Representational similarity modulates neural and behavioral signatures of novelty

Sophia Becker[1,2], Alireza Modirshanechi[1,2], and Wulfram Gerstner[1,2]

[1]School of Computer and Communication Sciences, EPFL
[2]School of Life Sciences, EPFL

May 1, 2024

Abstract

Novelty signals in the brain modulate learning and drive exploratory behaviors in humans and animals. Inherently, whether a stimulus is novel or not depends on existing representations in the brain, yet it remains elusive how stimulus representations influence novelty computation. In particular, existing models of novelty computation fail to account for the effects of stimulus similarities that are abundant in naturalistic environments and tasks. Here, we present a unifying, biologically plausible model that captures how stimulus similarities modulate novelty signals in the brain and influence novelty-driven learning and exploration. By applying our model to two publicly available data sets, we quantify and explain (i) how generalization across similar visual stimuli affects novelty responses in the mouse visual cortex, and (ii) how generalization across nearby locations impacts mouse exploration in an unfamiliar environment. Our model unifies and explains distinct neural and behavioral signatures of novelty, and enables theory-driven experiment design to investigate the neural mechanisms of novelty computation.

Introduction

Novelty signaling in the brain is known to facilitate learning [1–7], enhance sensory processing [8–11] and drive behaviour [12–17] in humans and animals encountering unfamiliar stimuli or situations. Computational models of novelty are valuable tools to explain and predict the effects of novelty signaling on neural activity and behaviour [14, 15, 18–21]. Existing computational novelty models often compute novelty as a decreasing function of the number of times a stimulus has been encountered previously ('count-based' novelty [18, 20, 22, 23], see [24] for a review). Some variations of these models consider only binary counts [14, 18], i.e. whether a stimulus is completely unfamiliar or not, while others base novelty computation on the stimulus frequency [15, 19, 21], i.e. the relative number of stimulus observations.

However, all these count-based novelty models rely on the assumption that stimuli can be precisely 'counted' and ignore any similarities between stimuli. This causes significant limitations in environments that are continuous or exhibit structured stimulus similarities. For example, consider two previously unobserved stimuli that are similar to each other but not identical, e.g. two paintings of mountain landscapes by the same artist that exhibit a similar painting technique and style. If the two stimuli are

presented sequentially, then how 'novel' is the second painting after observing the first one? Count-based novelty models force us to either (i) count the two stimuli separately, predicting that the second painting is completely novel despite its similarity to the first painting that was just observed; or (ii) count them as the 'same' stimulus, predicting that no matter whether we continue observing the first or the second painting, we get equally familiar with both of them. Both options are at odds with what we would expect from human and animal perception (see [25] for a review). In the same way, count-based novelty models fail to address novelty computation in continuous environments, e.g. during spatial exploration. Indeed, recent experiments suggest that the brain's novelty response to a given stimulus does not merely depend on the number of times the exact same stimulus has been observed [26–28], but is also modulated by exposure to specific features of parts of the stimulus [12, 29–31] (see Discussion). This underlines the conceptual limitations of count-based novelty models and the need to account for stimulus similarities in computational models of novelty.

In the field of machine learning, novelty-like signals are used to guide artificial agents during exploration of unfamiliar environments with sparse or no rewards [32–38]. Instead of using counts, these algorithms leverage neural networks to estimate the novelty of stimuli. Since, through training, the neural networks learn to generalize across similar stimuli, these methods of estimating novelty deal well with high-dimensional and continuous stimulus spaces. At the same time, however, machine learning-based novelty models rely on extensive training as well as architectures and learning rules with limited biological plausibility and interpretability.

Here, we propose a model of novelty computation that leverages probabilistic kernel-mixture models to combine the strengths of count-based and ML-inspired novelty models: our model is consistent with conventional count-based models in environments with discrete and distinct stimuli, but it also extends to continuous spaces and environments with similarity structure. Our model implements novelty updates as a biologically interpretable rule that is consistent with circuit models of novelty processing based on adaptation, Hebbian plasticity or inhibitory circuits [39–42]). Moreover, our model offers a way to test experimental hypotheses about how novelty-computing circuits represent stimuli and stimulus similarities.

Results

Count-based novelty models fail to account for similarity modulation of novelty responses

Most algorithmic models of novelty computation in the brain are based on estimating the counts [14, 18, 20, 22, 23] or the count-based frequency [15, 19, 21] at which stimuli (e.g., sensory stimuli, spatial locations etc.) have been observed: the more often or frequently a stimulus is observed, the less novel it is. These 'count-based' novelty models have four components: (i) an underlying stimulus space, (ii) a discretization of that stimulus space, (iii) an empirical count or frequency function that measures the familiarity of states, and (iv) a novelty function that computes the novelty of states based on their familiarity.

The stimulus space is determined by the environment and task for which novelty is computed. For example, in a task with Gabor-like visual stimuli (Fig. 1 A) that can be represented by their angular orientation, the stimulus space is the one-dimensional torus of possible Gabor orientations. In order to count stimuli in a stimulus space, we need to define a discretization. In general, in count-based novelty models, any two distinct stimuli s and s' are counted separately, while observations of two identical stimuli contribute to the same count. Examples where stimuli can be clearly identified as

‘distinct’ and ‘identical’ in this sense are human studies in abstract sequential decision-making tasks [15, 20, 43]. In continuous environments and for stimuli with continuous features like the orientation of Gabor filters, however, two stimuli are not necessarily fully ‘distinct’ or ‘identical’ – instead, they show continuously varying degrees of similarity. In these settings, counting is not well-defined. To still be able to count, we can bin the stimulus pace into equidistant bins (the bin width can e.g. be determined by psychometric discrimination thresholds [44, 45]). As we will elaborate in our toy example below, binning still fundamentally misses the continuous nature of stimulus similarities, i.e. the fact that two Gabor stimuli can be more or less similar (instead of ‘identical’ or ‘distinct’), depending on the difference between their angular orientations.

The third element of count-based novelty models is the empirical frequency function p that measures the familiarity of a stimulus s at time t:

$$p^{(t)}(s) = \frac{C^{(t)}(s)+1}{t+|S|}, \tag{1}$$

where $C_t(s)$ is the count of how often stimulus s has been observed up to the current time step t, and $|S|$ is number of available stimuli (or bins) S. At time $t = 0$, all stimuli start with the same initial frequency $p^{(0)}(s) = 1/|S|$. The frequency increases if a stimulus s is observed (by increasing the count C and time t), and decreases if it is not observed (by increasing only the time t). Some count-based models compute the state familiarity with simpler functions that, for example, consider absolute counts rather than state frequencies [18, 20, 22, 23] or only consider binary counts [14, 46]. Since these models can all be reformulated in terms of the empirical frequency p [24, 32], we focus on frequency-based novelty models as in Eq. 1 for the rest of the manuscript.

The fourth element of count-based novelty models is the novelty function N. The novelty of a stimulus s decreases nonlinearly with its familiarity:

$$N^{(t)}(s) = -\log p^{(t)}(s), \tag{2}$$

where the negative logarithm is a standard choice of nonlinearity in the novelty literature [24, 47]. The combination of empirical count-based frequency (Eq. 1) and logarithm (Eq. 2) makes the novelty decrease rapidly during the first few observations of a stimulus s, in line with experimental observations [40].

To illustrate the limitations of count-based novelty models, we consider a toy example (Fig. 1 A): we present three stimulus sequences with varying levels of stimulus similarities to a count-based novelty model. Each sequence consists of three Gabor stimuli with different angular orientations; the first and last stimuli are the same across sequences, but the second image has varying levels of similarity to the first image (sequence 1: identical, sequence 2: 15° difference, sequence 3: 30° difference). We evaluate the novelty response to each image as predicted by two count-based novelty models with different discretization (Fig. 1 B1-B2): the first one uses 180 small bins of 1° (corresponding to human perceptual acuity [44]), the second uses 4 large bins of 45°. Each Gabor images is represented by the average angular orientation of their bin; e.g., for the first discretization, any Gabor image with orientation between 44.5° and 45.5° is represented by $s = 45°$ etc. (Fig. 1 B1). We observe that count-based novelty with small bin size shows identical novelty responses to sequence 2 and 3 despite the different degree of similarity between the first two stimuli in sequence 2 and 3 (purple and blue lines in Fig. 1 C1): the novelty responses increase during the observation of the sequence since all stimuli are counted separately and the time t in Eq. 1 increases. For the first stimulus sequence (identical first two stimuli), count-based novelty with small bin size shows a significant decrease in novelty response the second time we observe the first stimulus (red line in Fig. 1 C1). Count-based novelty with large bin size, on the other hand, shows identical novelty responses to the first and the second sequence (red and purple lines in Fig. 1 C1-C2): The response to the second stimulus is equally attenuated in both cases. This is because count-based novelty with large bin size treats the second stimulus as identical to the first in both sequences, even though it is

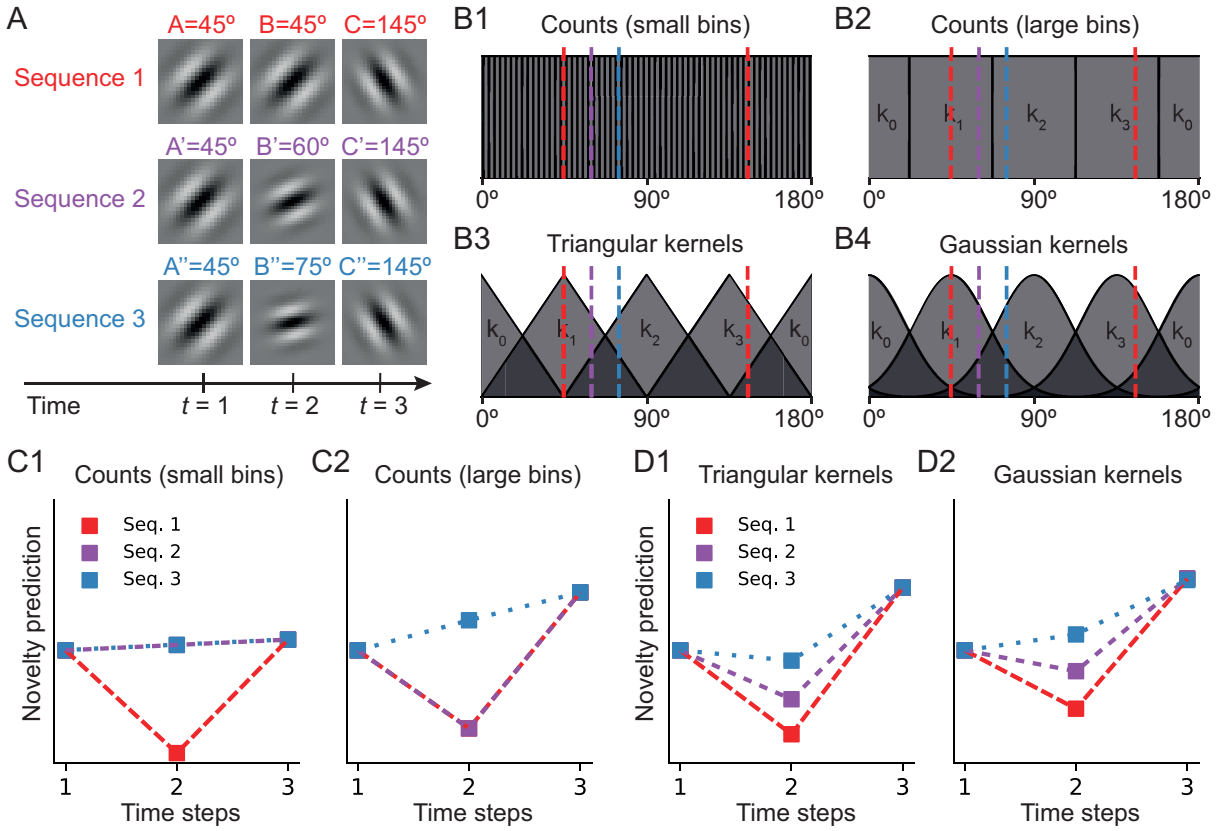


Figure 1: From count- to kernel-based novelty. **(A)** Toy example: three sequences with varying degrees of similarity between the first two stimuli in the sequence, respectively. Each sequence consists of three Gabor stimuli with different orientations, that are presented at consecutive time points. Stimuli are represented by their angular orientations between 0° and 180°. The first and third stimulus are the same across sequences, but the second stimulus shows varying levels of similarity to the first stimulus: 0° (identical) in seq. 1, 15° in seq. 2, 30° in seq. 3. **(B)** Count-based and kernel-based representations of the stimulus space of orientations ([0°, 180°]). Count-based novelty requires binning. Kernel-based novelty allows for a diverse range of kernel representations using (overlapping or non-overlapping) kernel functions. Whether two stimuli share a bin or kernels expresses whether and how these stimuli are similar, e.g. A' and B' are distinct for small bin sizes (B1), identical for large bin sizes (B2) and partially similar to varying degrees in the kernel representations (B3, B4). Dashed lines indicate the location of the second stimuli in seq. 1-3 from A (red: A=B and C from seq. 1, purple: B' from seq. 2, blue: B'' from seq. 3). **(C)** Novelty predictions for count-based novelty with either small bins (C1) or large bins (C2) for each of the three sequences in A. The models differ in their novelty predictions for the second stimulus in each sequence. Count-based novelty with small bins predicts identical novelty signals for seq. 2 and 3, even though they show varying degrees of similarity between the first two stimuli, respectively (15° in seq. 2 vs. 30° in seq. 3). Count-based novelty with large bins predicts identical novelty signals for seq. 1 and seq. 2, failing to capture the difference in similarity between the first two stimuli (0° in seq. 1 vs. 15° in seq. 2). **(D)** Novelty responses predicted by the kernel-based novelty models in B3-B4 for each stimulus in each of the sequences in A. Kernel-based novelty predictions reflect the different levels of similarity between the first two stimuli across the three sequences: the novelty predicted for the second stimulus increases from seq. 1 to seq. 3, with decreasing similarity between the first two stimuli.

significantly rotated (by 15° compared to the first stimulus) in sequence 2. For sequence 3, count-based novelty with large bin size (blue line in Fig. 1 C1) shows the same response as count-based novelty with small bin size: all stimuli are counted separately, such that the novelty responses increase throughout the sequence (with different slopes, depending on the bin size). Taken together, this exemplifies a fundamental problem of count-based novelty models: they do not account for stimulus similarities beyond a simple distinction between 'same' and 'different' – the bin size only allows us to choose where the distinction threshold is placed. This is in contradiction with common-sense intuition and experimental evidence [29–31] (see Discussion).

From counts to kernels: A generalized novelty model

To address the limitations of count-based novelty models, we propose a generalized novelty model that accounts for the effect of stimulus similarities and naturally extends to modeling novelty in continuous environments. Its core difference to count-based novelty models is the following: instead of computing the familiarity of stimuli as a function of their *discrete* empirical frequency (Eq. 1), our model uses kernel mixture models to compute a *continuous* empirical frequency density across the stimulus space S.

Kernel mixture models represent a probability density as the weighted sum of non-negative, normalized functions ('kernels') [48]. In our case, these kernels are defined over the stimulus space S. A kernel can have arbitrary shape, as long as it is non-negative and its integral over the state space is one. In general, each kernel can overlap with an arbitrary number of other kernels. In the context of our problem, kernels can serve as a flexible way to express stimulus similarities in both discrete and continuous spaces. For example, if the same kernel is activated by two different stimuli, these stimuli will influence each other's familiarity. Likewise, if the same stimulus activates two (overlapping) kernels, its familiarity will be influenced by other stimuli that also activate these kernels. In our toy example (Fig. 1 A), we can, e.g., choose overlapping triangular or Gaussian kernel to account for orientation similarities (Fig. 1 B3-B4). Note that kernel-based novelty with non-overlapping rectangular kernels is equivalent to count-based novelty with bins of the same width as the rectangular kernels (see Section 1.3 in the SI).

Based on a set of kernels on the stimulus space, we define a kernel mixture model p that generalizes the empirical frequency in Eq. 1. It quantifies the relative familiarity of each stimulus s at a given time point t as

$$p^{(t)}(s) = \sum_{j=1}^{N} w_j^{(t)} k_j(s), \tag{3}$$

where $k_1, ..., k_N$ are the kernel functions that account for the similarity structure of the stimulus space. The weights $w_1^{(t)}, ..., w_N^{(t)}$, with $\sum_{j=1}^{N} w_j^{(t)} = 1$, specify how much each kernel contributes to the empirical frequency density $p^{(t)}$ at a given time point t. In order for the empirical frequency density p to represent a meaningful and consistent notion of familiarity across space and time, we must choose the weights $w_j^{(t)}$ appropriately. We can do so using statistical inference: since we want our weights $w_j^{(t)}$ at any time point t to quantify the familiarity of different states based on the sequence of previous state observations $s_1, ..., s_t$, we choose the weights $w_j^{(t)}$ such that they maximize the likelihood of observing that state sequence (see Methods). Importantly, we show that these kernel weights can be updated with the following learning rule:

$$w_j^{(t)} = w_j^{(t-1)} + \alpha^{(t)} \left(\frac{k_j(s_t)}{p^{(t-1)}(s_t)} - 1 \right) w_j^{(t-1)}, \tag{4}$$

where $\alpha^{(t)} = \frac{1}{t+N}$ is the time-dependent learning rate. Importantly, the learning rule for the familiarity weights (Eq. 4) is *multiplicative*, similar to what has been suggested in a range of mechanistic models of novelty detection ([29, 39, 42, 49–51], see Discussion). Since the weight update in Eq. 4 only relies on information that is locally available in time, it can be implemented in a biologically plausible neural circuit model (see Methods and Section 1.4 in the SI).

Using the learning rule in Eq. 4 along with suitable kernel representations, we estimate the familiarity of a stimulus s at time t by its empirical frequency $p^{(t)}(s)$. Analogously to count-based novelty (Eq. 2), we then apply the negative logarithm to $p^{(t)}(s)$ to compute the (continuous) novelty function $N^{(t)}(s)$. The resulting kernel-based novelty model is (i) well-defined in discrete and continuous stimulus spaces, (ii) can account for similarities between stimuli via its kernel representation, and (iii) is equivalent to the established count-based novelty in discrete state spaces without similarity structure (SI, Section 1.3).

Kernel-based novelty generalizes across similar states

To illustrate the differences between count- and kernel-based novelty models, we compare the predictions of the two model types in our toy example (Fig. 1 A). Analogous to the count-based models discussed above (Fig. 1 B1-B2), we compute the predicted novelty responses to each stimulus in each sequence for two kernel-based novelty models with different state representations: (i) a representation with triangular kernels that encodes orientation similarities as linearly decreasing function of their angular difference (Fig. 1 B3), and (ii) a representation with Gaussian kernels similar to the tuning curves of orientation-selective cells in V1 (Fig. 1 B4).

We observe that the predictions of the kernel-based novelty models differ significantly from the predictions of the count-based model. Crucially, while count-based novelty predictions show identical responses either (i) to the first two sequences (large bins) or (ii) the last two sequences (small bins) and thus fail to capture stimulus similarities beyond fully 'distinct' and 'identical, kernel-based novelty predictions reflect the varying degree of similarity between the first two stimuli across the three sequences (Fig. 1 D1-D2): The novelty responses to the second stimulus in each sequence increase with decreasing levels of similarity (from sequence 1 to 3, red, purple and blue lines in (Fig. 1 D1-D2). This shows that, in contrast to count-based novelty, kernel-based novelty generalizes familiarity relative to stimulus similarity.

Kernel-based novelty explains novelty responses in mouse V1

In the previous section, we incorporated similarity-driven generalization into the computation of novelty signals using a kernel-based model. In this section, we ask whether our model can capture key features of experimentally measured novelty responses in the brain. To this end, we compare the novelty signals predicted by count- and kernel-based novelty models with neural responses to novelty in the mouse primary visual cortex (V1) during passive viewing of a sequence of 'familiar' and 'novel' images [39].

The images in the experiment of Homann et al. [39] each consist of multiple randomly placed and randomly oriented Gabor filters. Homann et al. study V1 novelty responses in mice during three variations of the experiment (Fig. 2 B1-B2), each illustrating a specific qualitative feature of novelty responses in V1. In the first version of the experiment, a deterministically repeating sequence of three images is presented for L repetitions ($L = 1, 3, 8, 18, 38$) before a 'novel' image is presented (Fig. 2 B1). Here, a 'novel' image is defined as an image that is not in the set of 'familiar' images that are shown as part of the repeated image sequence. The neural population response to the novel image increases with L (Fig. 3 A, black line). In the second variation of the experiment, the number M of familiar images in the sequence is varied ($M = 3, 6, 9, 12$; Fig. 2 B1), while the number of sequence repetitions L is fixed at $L = 18$. Increasing M decreases the neural responses to the novel image (Fig. 3 B1, black line) but does not affect the steady-state population activity after habituation to the familiar sequence (Fig. 3 B2, black line). The third variation of Homann et al.'s experiment investigates the recovery from familiarity: after neural responses to a familiar sequence ($M = 3$) have converged, a new sequence of images is presented for a number of repetitions L', before the initial sequence is shown again and the neural population response ΔN_{recov} is measured (Fig. 2 B2). Population responses to the formerly familiar sequence increase with L' (Fig. 3 C, black line).

To test whether these qualitative features of novelty responses in V1 can be captured by our kernel-based novelty model, we fit three different novelty models to the experimental data by Homann et al. (see Methods): a count-based novelty model, where are images are considered as distinct (equivalent to small bin sizes); a count-based model with larger bin sizes; and a kernel-based novelty model with triangular kernels. The bin widths (within the given range of 'small' and 'large' bins, see Methods) and the number of kernels are free parameters that are fitted together with the remaining parameters of each model (see

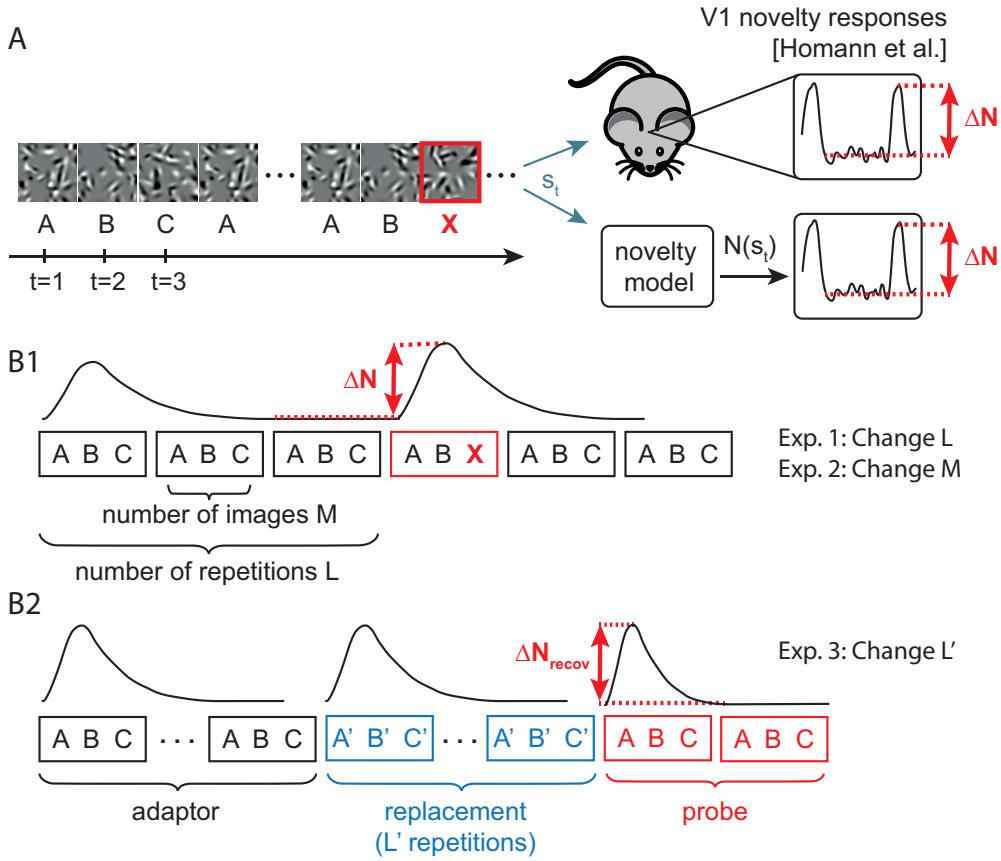


Figure 2: Passive viewing experiment: task and novelty model. (A) A sequence of M (here $M = 3$) consecutively presented visual stimuli is shown for L repetitions. In repetition $L+1$, the last element C of the sequence is replaced by a novel, i.e. previously unseen, stimulus X. For model simulations, the original stimuli used by Homann et al. (images with 100 randomly positioned and randomly oriented Gabor filters) are replaced with simplified stimuli that consists of a single, oriented Gabor filter with fixed location. We compare novelty signals ΔN (difference in responses to 'novel' and 'familiar' images) predicted by count-based and kernel-based novelty to the population responses of neurons in mouse V1 [39]. **(B1-B2)** Three variations of the passive viewing experiment in A: We consider responses to the 'novel' image (i) for different number of sequence repetitions L ($M = 3$ fixed; B1) and (ii) for different number M of 'familiar' images in a sequence ($L = 18$ fixed; B1); (iii) we assess the recovery from familiarity by first presenting an initial sequence ('adaptor') until novelty responses have converged, then presenting a second, different sequence for L' repetitions, and subsequently measuring the novelty response to the initially familiar sequence ('probe', $M = 3$ fixed; B2).

Methods). For the sake of simplicity and qualitative comparison, we use simplified stimuli in our model simulations: instead of overlapping Gabors, we use a single oriented Gabor filter with fixed location but randomly sampled orientation for each image (Fig. 2 A, see Methods). For each class of novelty model, we compare the fitted model with the neural data measured by Homann et al. (Fig. 3 A-C).

We find that all four features of V1 novelty responses are captured well by kernel-based novelty models with similarity-driven generalization (kernel novelty with triangular kernels, red line in Fig. 3 A-C) while count-based novelty fails to capture one or multiple features of V1 novelty responses (blue and purple lines in Fig. 3 A-C). This suggests that neural responses to novelty in mouse V1 are significantly modulated by generalization across similar stimuli. Specifically, while the recovery from familiarity is explained both by kernel-based novelty with triangular and count-based novelty with large bins (red and purple lines in Fig. 3 C), the decrease of novelty with increasing number of familiar images is captured only with triangular kernels (red line in Fig. 3 B1). Count-based novelty with large bins predicts a sudden decrease in novelty responses as the number M of familiar images in the sequence increases, which disagrees with the more gradual decline that is observed in the data. Count-based novelty with small bins, on the other hand, fails to capture both the decrease of novelty with the number of familiar images in the sequence (Fig. 3 B1) and the recovery from familiarity after prolonged replacement of a familiar stimulus sequence (Fig. 3 C). In particular, count-based novelty with small bins leads to a slower recovery from familiarity than the other novelty models. It also predicts a constant novelty response for an increasing number M of familiar images. Taken together, these findings suggest that (i) a quick recovery from familiarity when stimuli are not observed and (ii) a gradual decrease of novelty responses with increasing number of 'familiar' images are two essential features of V1 novelty computation that are significantly modulated by generalization across similar stimuli. The second feature, i.e. the decrease of responses with number of familiar images, is particularly susceptible to similarity-driven generalization, and can only be captured by kernel-based novelty with triangular kernels.

In addition to quantifying the effects of similarity-driven generalization on neural responses to novelty, kernel-based novelty also allows us to make qualitative hypotheses about the range of stimuli across which novelty computation generalizes in the primary visual cortex. For our specific representation, kernel-based models with a relatively small number (~ 5) of kernels best fit the neural data, while the best count-based discretization is to consider all stimuli as similar (Fig. 3 D). Overall, kernel-based novelty gives a better generalization of fit across the different variants of the experiment (Fig. 3 E).

Kernel-based novelty explains mouse exploration in an unfamiliar maze

In the previous section, we applied our kernel-based novelty model to a passive viewing task [39]. We now turn to behavioral correlates of novelty during active exploration of an unfamiliar, freely accessible maze [52]. Mice enter the maze through a single corridor that then separate into two corridors, which each branch separating again into two corridors etc. After six branching points, each corridor ends at a wall, where mice can only turn around and go back to the previous branching point. Since the branching points in the maze are organized in the structure of a binary tree (Fig. 4 C2), we refer to these end points where mice have to turn around as 'leaf nodes'. Half of the mice receive a water reward upon licking a reward port in one of the maze's leaf nodes ('goal state'). To assure that we only include novelty-driven behaviour in our analysis, we only consider mouse behaviour on the path between their first entry into the labyrinth and their first encounter with the goal state. During this exploration phase, mice from the rewarded and the unrewarded group have access to the same kind of information and do not show significant behavioural differences [52], such that we treat them equivalently in our analysis.

We model the mouse behaviour using novelty-seeking reinforcement learning (RL) models that maximize intrinsically computed novelty [15, 18, 25, 32, 53] (also see [54–60] for similar approaches using different intrinsic motivations) instead of extrinsic rewards from the environment (Fig. 4 B, see Meth-

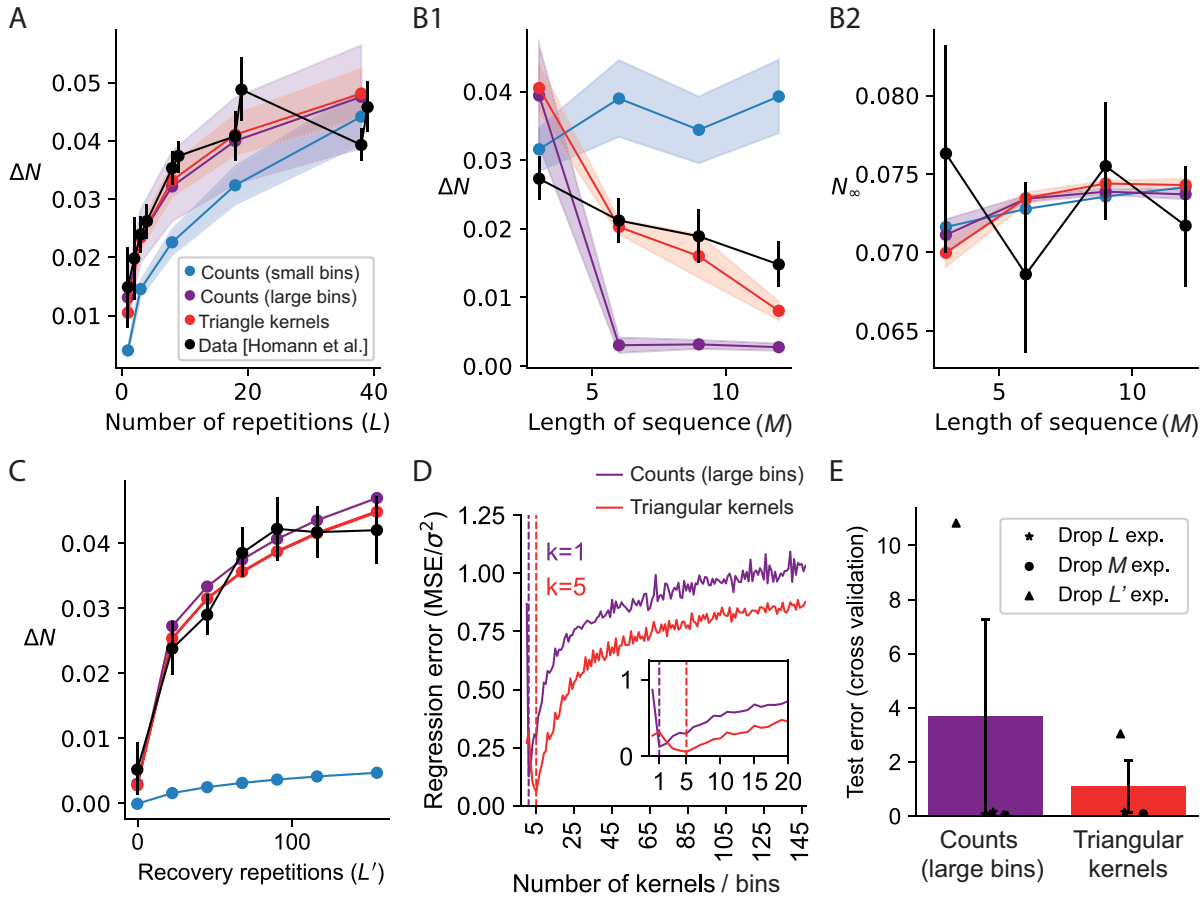


Figure 3: Kernel-based novelty explains novelty responses in mouse V1. (A-C) V1 novelty responses ΔN [39] (black line) and novelty predictions ΔN of fitted count-based (blue line: small bin size, purple line: large bin size) and kernel-based novelty models (red line) during the three passive viewing experiments in Fig. 2: Different number of repetitions L of the familiar sequence (A), different number M of familiar images per sequence (B1-B2), and recovery of novelty responses after L' repetitions of a different stimulus sequence (C). For the variable-M experiment, both the response to the 'novel' image (ΔN, B1) and the steady state response to the 'familiar' sequence (N_∞, B2) are shown. Kernel-based novelty with triangular kernels reproduces all qualitative features of the novelty responses in V1 (red line). Count-based novelty with large bins (purple line) lead to quicker decline of novelty responses with M than neural data (B1). Novelty predictions of count-based novelty with small bins (blue line) do not match neural novelty responses, in particular with respect to the recovery from familiarity (C) and the change of novelty responses with M (B1). This suggests that these two features of neural novelty responses rely on similarity-driven generalization. **(D)** Normalized regression error between novelty prediction and neural data for different number of kernels or bins K (remaining parameters fitted to neural data), for each novelty model. Best performance (low regression error) is achieved for about five kernels with the triangular kernel model and a single box with the binning (box kernel) model, indicating that the best explanation in a count-based model is to consider all stimuli as identical. **(E)** Cross-validation error between model predictions and neural responses in each of the three experiments, where models were fitted to the neural data from the respective other experiments. Kernel-novelty with triangular kernels has lower test error, i.e. generalizes better across experiments than count-based novelty with large bins.

ods). In order to compare the exploration behaviour of different 'agents', i.e. mice and novelty-seeking RL models, we define 'states' and 'actions' in the labyrinth. States include the home cage, the branching points and the leaf nodes (green circles in Fig. 4 A). In each of the branching points, mice or RL agents can take one of three actions (purple arrows in Fig. 4 A): (i) going down the left corridor until the next branching point or leaf node, (ii) going down the right corridor until the next branching point or leaf node, or (iii) going back to the previous branching point. In the home cage and the leaf nodes, agents only have one available action, i.e. going back.

We compare mouse behavior to three families of novelty-seeking RL models, that are based on either

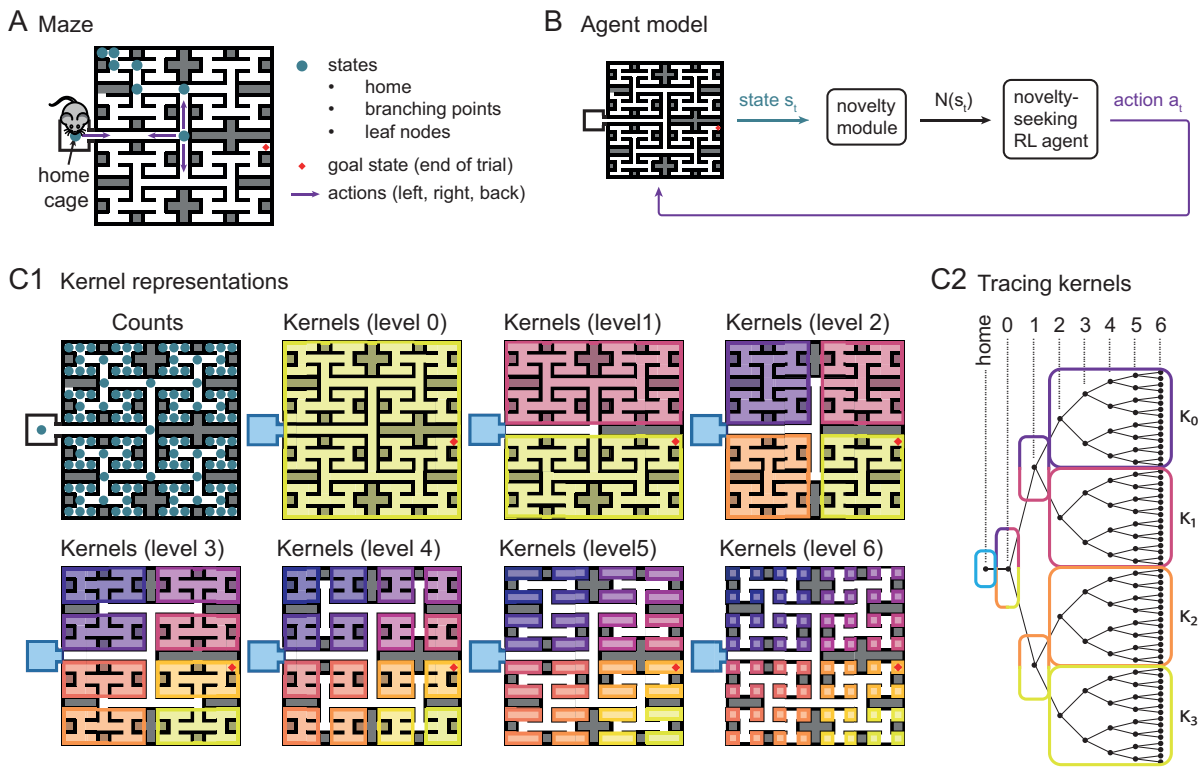


Figure 4: Exploration behaviour of mice and novelty-seeking reinforcement learning (RL) agents. We model the exploration behaviour of mice in a labyrinth [52], using novelty-seeking RL agents. **(A)** The maze of Rosenberg et al. [52]. To compare the behaviour of mice and novelty-seeking RL agents, we define states and actions. The states (blue dots, example states on a short trajectory), with respective actions available in each state, are (i) the home cage (single action: go forward into the maze), the 'branching points' of the labyrinth (three available actions: go left, go right or go backwards), and the 'leaf nodes' of the labyrinth where mice hit the labyrinth wall (single action: go backwards). Half of the mice in the experiment by Rosenberg et al. receive a water reward in one of the leaf nodes (red diamond), but since we are interested in intrinsically driven exploration, we only consider mouse behaviour up to their first encounter of this 'goal' state. **(B)** In each time step t, a novelty-seeking RL agent takes its current position in the labyrinth (state s_t) as input and computes the state novelty using its internal novelty model. Then it updates its internal novelty value function and policy based on a given RL algorithm (either model-free, model-based or hybrid, see Methods), and chooses its next action a_t in the labyrinth from its policy. **(C1-C2)** State representations used for the RL agents' internal novelty models. (C1, first panel) State representation for RL agents with count-based novelty (each state separate). (C1, remaining panels) Kernel functions that tile the state space into 'areas' of the maze, e.g., into quadrants in the 'level 2' kernel representation. Each kernel contains all states in a given 'area' of the maze (marked in different colors) as well as the states along the shortest path between the home cage and its encoding area (C2). For example, the purple 'level 2' kernel contains all states within the purple quadrant as well as the two branching point states that connect it to the home cage. Each representation encodes the labyrinth with a different a level of 'granularity', e.g. level 0 only distinguishes the home cage and the rest of the maze while level 6 distinguishes each individual branching point or leaf node (together with the corresponding trace back to the home cage). (C2) Sketch of the 'level 2' kernels in the binary tree representation of the maze; each node is a state and two tree nodes are connected if the corresponding states in the labyrinth are connected via a corridor.

model-free (MF), model-based (MB) or hybrid (Hyb) RL algorithms (see Methods). To compute the novelty signals for each RL model, we use either count-based novelty, which is not modulated by generalization across neighboring states, or kernel-based novelty, which generalizes across states in the maze according to its underlying kernel representation. To assess the impact of the kernel representation on novelty-driven exploration, we compare seven kernel-based novelty models whose kernel representations differ in how they generalize novelty across nearby states in the maze (Fig. 4 C1-C2, see Methods). The kernels we define are inspired by the notion that mice, while exploring, utilize information about (i) how they got from the home cage to their current location (shortest path between a state s and the home state), and (ii) which 'area' of the maze they are in (e.g., a given quadrant of the labyrinth for the

‘level 2’ representation in Fig. 4 C2). We therefore also refer to these kernels as ‘tracing kernels’ The kernel representations differ with respect to how large the ‘areas’ are that their kernel functions encode. For example, the ‘level 2’ representation has kernels that generalize across all states in one quadrant, respectively, while the ‘level 5’ kernels generalize across the two closest branching points of a given leaf node (Fig. 4 C1). We refer to this as the ‘granularity level’ of the kernel representation: representations with higher granularity have more and smaller kernels, while representations with lower granularity have fewer but broader kernels.

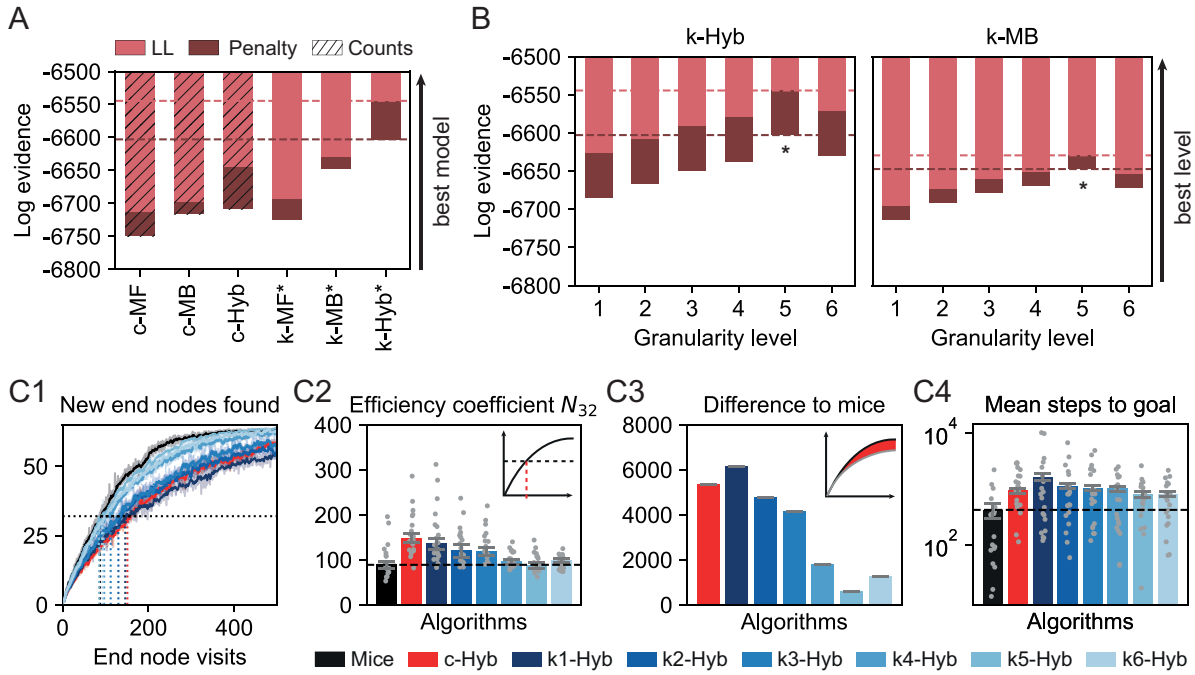


Figure 5: RL agents driven by kernel-based novelty explain the exploration behaviour of mice in the labyrinth. (A) Model fit of different novelty-seeking RL models to mouse behaviour during labyrinth exploration. We compare novelty-seeking RL models with (i) different types of RL algorithms (model-free: MF, model-based: MB, hybrid: Hyb) and (ii) different novelty models (count-based: prefix c-, kernel-based: prefix k-). For agents with kernel-based novelty, we report the model performance for the best state representation (indicated by *), i.e. the best ‘granularity level’ (see Fig. 4 C1, Methods). Novelty-seeking agents with kernel-based novelty show a better fit to mouse behaviour than agents with count-based novelty. The winning model family is the hybrid RL agent with kernel-based novelty, followed by MB agents with kernel-based novelty. **(B)** Model fit for hybrid and model-based agents with kernel-based novelty, for state representations with different levels of granularity. In both model families, state representations with level 5-granularity achieve the best fit to mouse behaviour. **(C1-C4)** As posterior predictive checks of the model comparison results in A-B, we compare mouse and model behaviour with respect to different statistics: (C1) the mean number of new end nodes found, as a function of the total number of visits to leaf nodes; (C2) the mean number of leaf node visits until half of the 64 end nodes are found (efficiency coefficient N_{32}); (C3) the integrated difference between the end node discovery curves of agents and mice in C1; and (C4) the median number of steps that mice or agents take to reach the ‘goal state’ for the first time. Consistent with the model comparison results, hybrid agents with kernel-based novelty and level 5-state representation match mouse behaviour best with respect to these statistics.

To test whether exploration driven by kernel-based novelty with ‘tracing kernels’ helps explaining mouse behaviour, we fit all of the model variants described above (Fig. 4 C1), i.e. all combinations of RL models (MF, MB, Hyb) and novelty models (count-based, kernel-based with different state representations) to mouse behaviour, using maximum likelihood estimation of the parameters (see Methods). To determine which model family explains the data best, we then perform a Bayesian model comparison between the best-performing models of each family (see Methods). Across all types of RL models, exploration driven by kernel-based novelty gives a better fit to mouse behaviour than exploration driven by count-based novelty (Fig. 5 A). We further investigate which kernel representation gives an exploration most similar to that of mice. To this end, we perform a model comparison between all models from the best two RL

families (hybrid and model-based) that maximize kernel-based novelty with state representations of different granularity (see Methods). We find that kernel-based novelty models with state representations that generalize across small neighborhoods (high but not maximal granularity) best explain the exploration behaviour of mice (Fig. 5 B): for both hybrid and model-based RL families, the winning model uses kernel-based novelty with level-5 granularity, i.e. a state representation that generalizes across two immediately neighboring leaf nodes and their shared branching point (Fig. 5 B).

To validate our model comparison results and analyze the behaviour of the models that best capture mouse exploration, we perform posterior predictive checks [61] (see Methods). In particular, we simulate count-based and the different kernel-based novelty-driven RL models and compare them to mice, with respect to different statistics that assess the efficiency and other features of their exploration behaviour (Fig. 5). As expected, the winning model (kernel-based novelty-driven model with level-5 granularity) approximates mouse behaviour best with respect to all statistics, and shows the best exploration efficiency among all tested models (Fig. 5). In contrast, both the absence of generalization (count-based novelty) and over-generalization (kernel-based novelty with 'lower level' representations) show significantly lower exploration efficiency than mice and the winning model, confirming our previous model comparison results (Fig. 4). Taken together, these findings suggest that generalization of familiarity across nearby or recently experienced ('trace') states could significantly modulate spatial exploration of mice in unfamiliar environments.

Discussion

We developed a generalized model of novelty computation that leverages kernel mixture models to capture the effect of stimulus similarities on novelty computation and extends current computational novelty models to continuous state spaces. In tasks where all stimuli are discrete and distinct, our kernel-based novelty model is equivalent to count-based novelty [14,15,18–23,46]. We show that kernel-based novelty captures novelty responses in mouse V1 [39] that are unexplained by count-based models. By combining kernel-based novelty with intrinsically motivated reinforcement learning models [15, 18, 25, 32, 53], we further show that seeking kernel-based novelty captures the exploration behavior of mice [52] better than seeking count-based novelty.

Our results suggest that representational similarities between unfamiliar and familiar stimuli significantly influence both neural and behavioral signatures of novelty-related processing in the brain. Specifically, (i) high similarity to familiar stimuli attenuates the novelty response amplitudes to a previously unobserved stimuli, and (ii) generalization of familiarity across nearby spatial locations can make exploration more efficient by reducing the time spent exploring 'similar' (=close-by) regions in the maze. Our model unifies these findings by showing that they can both arise due to the modulation of novelty signals by stimulus similarities. Building on this, we predict that also novelty responses in other sensory modalities [4, 11, 30, 62] and in the hippocampus [6, 7, 10] as well as downstream novelty signaling in salience-related regions [5, 16, 17, 31, 63] could be subject to similarity modulation. If this prediction is true, it would have important implications for the experimental study of novelty: 'novel' input stimuli should either be chosen as almost perfectly distinct (e.g. [15, 31, 64]) to eliminate the impact of stimulus similarities on novelty signals, or their similarity should be controlled for as an experimental variable that can have significant influence on novelty signals and novelty-related behavior. Apart from allowing for theory-driven experimental hypotheses and experiment design, our kernel-based novelty model also allows us to reinterpret existing experimental findings in the context of similarity-driven generalization of familiarity. For example, a study by Montgomery [12] that investigates the impact of luminance on mice exploration in otherwise identical mazes finds that a larger luminosity difference between two successively explored mazes (i.e. less similarity) leads to more exploration in the second maze. This relationship is a signature of exploration driven by similarity-modulated novelty. DeBaene and Vogels [29]

report that neuronal adaptation in the monkey IT cortex significantly depends on the difference between adaptor and test stimulus, with respect to features such as object shape and location. This links directly to possible circuit implementations through which stimulus similarities could modulate novelty signals.

In addition to experimental evidence, insights from machine learning support the functional need to generalize novelty information across similar stimuli [25]. In particular, Jaegle et al. [25] argue that any system that computes novelty has to solve the problems of (i) mapping a diversity of high-dimensional inputs to reasonable states that abstract away e.g. different views on the same object ('view invariance'), and (ii) grouping states together based on their shared characteristics ('state invariance'). Kernel-based novelty solves the second problem by expressing similarities in a given representation space by (potentially overlapping) kernel functions. In this work, we address the first challenge, i.e. the choice of an appropriate state representation, by constructing stimulus representations based on experimental knowledge and hypotheses. However, kernel-based novelty is not limited to such 'experimentally justified' representations and can also be used in the latent space of trained deep networks, e.g. deep networks with 'brain-like' behavior [65, 66]. Some machine learning algorithms solve both view and state invariance problem simultaneously by training a deep network to a estimate familiarity density from which novelty can be computed via pseudo-counts [32, 33]. Others use hashing [34] to map similar states to discrete bins that can then be used with count-based novelty (see [67] for an interesting neural implementation), or construct meaningful latent spaces where the novelty of a state is defined as a function of how many other states are within a given Euclidean distance in the latent space [38]. These methods are powerful since they learn an appropriate state representation simultaneously with the familiarity density (given an appropriate network architecture and loss function). However, in contrast to kernel-based novelty, they rely on backpropagation to estimate the familiarity density. Since machine-learning-based novelty models are usually trained end-to-end, they are not compatible with hand-designed representation based on 'experimental knowledge', making it harder to investigate how a certain distribution of place fields influence novelty-driven exploration, or how novelty signals in V1 depend on the specific tuning of their cells.

One of the central advantages of kernel-based novelty is its biologically interpretable (local) learning rule. Specifically, the kernel weights that capture the empirical frequency of each kernel are updated with a multiplicative learning rule. While kernel-based novelty does not make explicit predictions about which circuit structure has to underlie novelty computation, its update rule for the novelty weights is consistent with several types of circuit mechanisms proposed for novelty computation, including input adaptation and short-term synaptic depression [29, 39, 42, 49]), inhibitory circuits [41], or Hebbian or anti-Hebbian multiplicative plasticity [50, 51].

In conclusion, we propose a generalized computational framework to model novelty computation in the brain that allows to assess the impact of similarity-driven generalization on neural novelty responses and novelty-driven behavior. Our approach opens new possibilities for future theory-driven experiment design.

Acknowledgements

S.B. would like to thank Martin Barry for helpful discussions about possible circuit implementations of our model. This work was supported by the Swiss National Science Foundation No. 200020_207426.

References

[1] Gregor Rainer and Earl K. Miller. Timecourse of object-related neural activity in the primate prefrontal cortex during a short-term memory task. *European Journal of Neuroscience*, 15:1244–1254, 2002.

[2] John E. Lisman and Anthony A. Grace. The hippocampal-VTA loop: controlling the entry of information into long-term memory. *Neuron*, 46:703–713, 2005.

[3] Tomonori Takeuchi, Adrian J. Duszkiewicz, Alex Sonneborn, Patrick A. Spooner, Miwako Yamasaki, Masahiko Watanabe, Caroline C. Smith, Guillén. Fernández, Karl Deisseroth, Robert W. Greene, and Richard G. M. Morris. Locus coeruleus and dopaminergic consolidation of everyday memory. *Nature*, 537:357–362, 2016.

[4] Joachim Morrens, Çağatay Aydin, Aliza Janse van Rensburg, José Esquivelzeta Rabell, and Sebastian Haesler. Cue-evoked dopamine promotes conditioned responding during learning. *Neuron*, 106(1):142 – 153.e7, 2020.

[5] Munir G. Kutlu, Jennifer E. Zachry, Patrick R. Melugin, et al. Dopamine signaling in the nucleus accumbens core mediates latent inhibition. *Nature Neuroscience*, 25:1071–1081, 2022.

[6] James B. Priestley, John C. Bowler, Sebi V. Rolotti, Stefano Fusi, and Attila Losonczy. Signatures of rapid plasticity in hippocampal CA1 representations during novel experiences. *Neuron*, 110:1978–1992, 2022.

[7] Ruy Gomez-Ocadiz, Massimiliano Trippa, Chun-Lei Zhang, and et al. A synaptic signal for novelty processing in the hippocampus. *Nature Communications*, 13, 2022.

[8] Charan Ranganath and Gregor Rainer. Neural mechanisms for detecting and remembering novel events. *Nature Reviews Neuroscience*, 4:193–202, 2003.

[9] Nico Bunzeck and Emrah Düzel. Absolute coding of stimulus novelty in the human substantia nigra/VTA. *Neuron*, 51:369–379, 2006.

[10] Emrah Düzel, Nico Bunzeck, Marc Guitart-Masip, and Sandra Düzel. Novelty-related motivation of anticipation and exploration by dopamine (NOMAD): implications for healthy aging. *Neuroscience & Biobehavionral Reviews*, 34:660–669, 2010.

[11] Nicholas C. Foley, David C. Jangraw, Christopher Peck, and Jacqueline Gottlieb. Novelty enhances visual salience independently of reward in the parietal lobe. *Journal of Neuroscience*, 34(23):7947–7957, 2014.

[12] K. C. Montgomery. Exploratory behavior as a function of 'similarity' of stimulus situations. *Journal of Comparative and Physiological Psychology*, 46:129–133, 1953.

[13] Daniel E. Berlyne. Curiosity and exploration. *Science*, 153:25–33, 1966.

[14] Irene C. Dezza, Xavier Noel, Axel Cleeremans, and Angela J. Yu. Distinct motivations to seek out information in healthy individuals and problem gamblers. *Translational Psychiatry*, 11:408, 2021.

[15] He A. Xu, Alireza Modirshanechi, Marco P. Lehmann, Wulfram Gerstner, and Michal H. Herzog. Novelty is not Surprise: Human exploratory and adaptive behavior in sequential decision-making. *PLoS Computational Biology*, 17:e1009070, 2021.

[16] Akiti Korleki, Iku Tsutsui-Kimura, Yudi Xie, Alexander Mathis, Jeffrey E. Markowitz, Rockwell Anyoha, Sandeep Robert Datta, Mackenzie Weygandt Mathis, Naoshige Uchida, and Mitsuko Watabe-Uchida. Striatal dopamine explains novelty-induced behavioral dynamics and individual variability in threat prediction. *Neuron*, 110:3789–3804, 2022.

[17] Takaya Ogasawara, Fatih Sogukpinar, Kaining Zhang, Yang-Yang Feng, Julia Pai, Ahmad Jezzini, and Ilya E. Monosov. A primate temporal cortex–zona incerta pathway for novelty seeking. *Nature Neuroscience*, 25:50–60, 2022.

[18] Samuel J. Gershman and Yael Niv. Novelty and inductive generalization in human reinforcement learning. *Topics in cognitive science*, 7(3):391–415, 2015.

[19] Maxime Maheu, Stanislas Dehaene, and Florent Meyniel. Brain signatures of a multiscale process of sequence learning in humans. *Elife*, 8:e41541, 2019.

[20] Rachit Dubey and Thomas L Griffiths. Reconciling novelty and complexity through a rational analysis of curiosity. *Psychological Review*, 127(3):455–476, 2020.

[21] Alireza Modirshanechi, He A. Xu, Wei-Hsiang Lin, Michael H. Herzog, and Wulfram Gerstner. The curse of optimism: a persistent distraction by novelty. *bioRxiv*, 2022.

[22] Sham Kakade and Peter Dayan. Dopamine: generalization and bonuses. *Neural Networks*, 15(4-6):549–559, 2002.

[23] Alexander L. Strehl and Michael L. Littman. An analysis of model-based interval estimation for Markov decision processes. *J Comput Syst Sci*, 74:1309–1331, 2008.

[24] Alireza Modirshanechi, Sophia Becker, Johanni Brea, and Wulfram Gerstner. Surprise and novelty in the brain. *Current Opinion in Neurobiology*, 82:102758, 2023.

[25] Andrew Jaegle, Vahid Mehrpour, and Nicole Rust. Visual novelty, curiosity, and intrinsic reward in machine learning and the brain. *Current Opinion in Neurobiology*, 58:167–174, 2019.

[26] D. Berlyne, I. Koenig, and T. Hirota. Novelty, arousal, and the reinforcement of diversive exploration in the rat. *Journal of Comparative and Physiological Psychology*, 62:222–226, 1966.

[27] Earl K. Miller and Robert Desimone. Parallel neuronal mechanisms for short-term memory. *Science*, 263, 1994.

[28] J.-Z. Xiang and M. W. Brown. Differential neuronal encoding of novelty, familiarity and recency in regions of the anterior temporal lobe. *Neuropharmacology*, 37:657–676, 1998.

[29] Wouter De Baene and Rufin Vogels. Effects of adaptation on the stimulus selectivity of macaque inferior temporal spiking activity and local field potentials. *Cerebral Cortex*, 20:2145–2165, 2009.

[30] Travis Meyer and Nicole C. Rust. Single-exposure visual memory judgments are reflected in inferotemporal cortex. *Elife*, 7:e32259, 2018.

[31] Kaining Zhang, Ethan S. Bromberg-Martin, Fatih Sogukpinar, Kim Kocher, and Ilya E. Monosov. Surprise and recency in novelty detection in the primate brain. *Current Biology*, 32(10):2160–2173.e6, 2022.

[32] Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*, pages 1471–1479, 2016.

[33] Georg Ostrovski, Marc G. Bellemare, Aäron van den Oord, and Rémi Munos. Count-based exploration with neural density models. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2721–2730. PMLR, 06–11 Aug 2017.

[34] Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. #Exploration: A Study of Count-Based Exploration for Deep Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[35] Jarryd Martin, S Suraj Narayanan, Tom Everitt, and Marcus Hutter. Count-based exploration in feature space for reinforcement learning. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 2471–2478. AAAI Press, 2017.

[36] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. *arxiv*, 2018.

[37] Nikolay Savinov, Anton Raichuk, Raphaël Marinier, Damien Vincent, Marc Pollefeys, Timothy Lillicrap, and Sylvain Gelly. Episodic curiosity through reachability. *ICLR*, 2019.

[38] Ruo Yu Tao, Vincent François-Lavet, and Joelle Pineau. Novelty search in representational space for sample efficient exploration. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 8114–8126. Curran Associates, Inc., 2020.

[39] Jan Homann, Sue A. Koay, Kevin S. Chen, David W. Tank, and Michael J. Berry. Novel stimuli evoke excess activity in the mouse primary visual cortex. *Proceedings of the National Academy of Sciences*, 119:e2108882119, 2022.

[40] Rafal Bogacz and Malcolm W. Brown. Comparison of computational models of familiarity discrimination in the perirhinal cortex. *Hippocampus*, 13:494–524, 2003.

[41] Auguste Schulz, Christoph Miehl, Michael J. Berry II, and Julijana Gjorgjieva. The generation of cortical novelty responses through inhibitory plasticity. *Elife*, 10:e65309, 2021.

[42] Kyle Aitken, Luke Campagnola, Marina Garrett, Shawn Olsen, and Stefan Mihalas. Familiarity modulated synapses model visual cortical circuit novelty responses. *bioRxiv*, 2023.

[43] Nathaniel D Daw, Samuel J Gershman, Ben Seymour, Peter Dayan, and Raymond J Dolan. Model-based influences on humans' choices and striatal prediction errors. *Neuron*, 69(6):1204–1215, 2011.

[44] William H. A. Beaudot and Kathy T. Mullen. Orientation discrimination in human vision: Psychophysics and modeling. *Vision Research*, 46:26–46, 2006.

[45] Lupeng Wang, Kerry McAlonan, Sheridan Goldstein, Charles R. Gerfen, and Richard J. Krauzlis. A causal role for mouse superior colliculus in visual perceptual decision-making. *Journal of Neuroscience*, 40:3768–3782, 2020.

[46] Samuel J Gershman, Marie-H Monfils, Kenneth A Norman, and Yael Niv. The computational nature of memory modification. *Elife*, 6:e23763, 2017.

[47] Andrew Barto, Marco Mirolli, and Gianluca Baldassarre. Novelty or surprise? *Frontiers in psychology*, 4:907, 2013.

[48] Christopher M. Bishop and Nasser M. Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[49] Robert Mill, Martin Coath, Thomas Wennekers, and Susan L. Denham. A neurocomputational model of stimulus-specfic adaptation to oddball and markov sequences. *PLoS Computational Biology*, 7:e1002117, 2011.

[50] Sukbin Lim, Jillian L. McKee, Luke Woloszyn, Yali Amit, David J. Freedman, David L. Sheinberg, and Nicolas Brunel. Inferring learning rules from distributions of firing rates in cortical neurons. *Nature Neuroscience*, 18:1804–1810, 2015.

[51] Krithika Mohan, Ulises Pereira-Obilinovic, Stanislav Srednyak, Yali Amit, Nicolas Brunel, and David Freedman. Visual familiarity learning at multiple timescales in the primate inferotemporal cortex. *bioRxiv*, 2024.

[52] Matthew Rosenberg, Tony Zhang, Pietro Perona, and Markus Meister. Mice in a labyrinth show rapid learning, sudden insight, and efficient exploration. *eLife*, 10:e66175, 2021.

[53] Nuttapong Chentanez, Andrew G Barto, and Satinder P Singh. Intrinsically motivated reinforcement learning. In *Advances in neural information processing systems*, pages 1281–1288, 2005.

[54] Adrien Baranes and Pierre-Yves Oudeyer. Active learning of inverse models with intrinsically motivated goal exploration in robots. *Robotics and Autonomous Systems*, 61.

[55] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 16–17, 2017.

[56] Arthur Aubret, Laetitia Matignon, and Salima Hassas. An information-theoretic perspective on intrinsic motivation in reinforcement learning: A survey. *Entropy*, 25(2), 2023.

[57] Dmytro Grytskyy, Jorge Ramírez-Ruiz, and Rubén Moreno-Bote. A general markov decision process formalism for action-state entropy-regularized reward maximization. *arxiv*, 2023.

[58] Rubén Moreno-Bote and Jorge Ramirez-Ruiz. Empowerment, free energy principle and maximum occupancy principle compared. In *NeurIPS 2023 workshop: Information-Theoretic Principles in Cognitive Systems*, 2023.

[59] Jorge Ramirez-Ruiz, Dmytro Grytskyy, Chiara Mastrogiuseppe, Yamen Habib, and Ruben Moreno-Bote1. Complex behavior from intrinsic motivation to occupy future action-state path space. *arxiv*, 2024.

[60] Franziska Brändle, Lena J. Stocks, Joshua B. Tenenbaum, Samuel J. Gershman, and Eric Schulz. Empowerment contributes to exploration behaviour in a creative video game. *Nature Human Behavior*, 7, 2023.

[61] Andrew Gelman, John B. Carlin, Hal S. Stern, David B. Dunson, Aki Vehtari, and Donald B. Rubin. *Bayesian Data Analysis*. Chapman and Hall/CRC, 3 edition, 2013.

[62] Itai Hershenhoren, Nevo Taaseh, Flora M. Antunes, and Israel Nelken. Intracellular correlates of stimulus-specific adaptation. *The Journal of Neuroscience*, 34:3303–3319, 2014.

[63] Ilya E. Monosov. Curiosity: primate neural circuits for novelty and information seeking. *Nature Reviews Neuroscience*, 25, 2024.

[64] A. Ghazizadeh and O. Hikosaka. Salience memories formed by value, novelty and aversiveness jointly shape object responses in the prefrontal cortex and basal ganglia. *Nature Communications*, 13, 2022.

[65] Dan Yamins and James J. DiCarlo. Using goal-driven deep learning models to understand sensory cortex. *Nature Neuroscience*, 19, 2016.

[66] Martin Schrimpf, Jonas Kubilius, Michael J. Lee, N. Apurva Ratan Murty, Robert Ajemian, and James J. DiCarlo. Integrative benchmarking to advance neurally mechanistic models of human intelligence. *Neuron*, 108, 2020.

[67] Sanjoy Dasgupta, Daisuke Hattori, and Saket Navlakha. A neural theory for counting memories. *Nature Communications*, 13:5961, 2022.

[68] S. J. Nowlan. *Soft Competitive Adaptation: Neural Network Learning Algorithms based on Fitting Statistical Mixtures*. PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, 1991.

[69] Geoffrey J. McLachlan and Thriyambakam Krishnan. *The EM Algorithm and Extensions*. 2007.

[70] Matthew Rosenberg, Tony Zhang, Pietro Perona, and Markus Meister. Rosenberg-2021-repository. `https://github.com/markusmeister/Rosenberg-2021-Repository`, 2021.

[71] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Methods

Update rule for kernel-based novelty

Normative approach. The central element of kernel-based novelty is the kernel-mixture model that defines the familiarity density p (Eq. 3) and whose weights at time point t are chosen as the maximum-a-posteriori (MAP) estimate of the sequence $s_{1:t}$ of stimulus observations up to t:

$$\mathbf{w}^{(t)} = \arg\max_{\mathbf{w}} \log \Pr(s_{1:t}|\mathbf{w}) \Pr(\mathbf{w}), \tag{5}$$

where $\mathbf{w}^{(t)} = \left(w_1^{(t)}, ..., w_N^{(t)}\right)^{\mathrm{T}}$ is the vector of kernel mixture weights at time t, and $\Pr(\mathbf{w})$ is a Dirichlet prior over the weights (see SI). We solve Eq. 5 using the incremental expectation-maximization (EM) algorithm [48, 68, 69]. The resulting closed-form solution for the weights can then be reformulated into the delta learning rule given in the main text (Eq. 4). First, we outline the final update rules that we derived by applying the incremental EM algorithm to our model (all details of the derivation are provided in the SI).

EM updates. For each time point t at which we observe a stimulus s_t, we compute the E-step (estimation step) of the incremental EM algorithm, i.e. we compute 'responsibilities' $\gamma_{j,t}$ that estimate how well a given kernel k_j of the mixture distribution at time $t-1$ captures the observation s_t:

$$\gamma_{j,t} = \frac{w_j^{(t-1)} k_j(s_t)}{\sum_{j'=1}^{N} w_{j'}^{(t-1)} k_{j'}(s_t)}. \tag{6}$$

The initial weights are chosen uniformly as $w_j^{(0)} = \frac{1}{N}$ for all $j = 1, ..., N$. In the subsequent M-step (maximization step) for time point t, we compute the new weights $w_j^{(t)}$ based on the responsibilities $\gamma_{j,t'}$, $j = 1, ..., N$, $t' = 1, ..., t$ that were estimated in all previous E-steps (Eq. 6):

$$w_j^{(t)} = \frac{1}{t + \varepsilon(N-1)} \sum_{t'=1}^{t} \gamma_{j,t'}. \tag{7}$$

Iterative weight update rule. Note that the expression for the mixture weight at time t (Eq. 7) still depends on the information from previous E-steps through the 'responsibilities' $\gamma_{j,t'}$, $t' = 1, ..., t-1$. However, we can reformulate Eq. 7 to obtain an iterative weight update rule that computes $\mathbf{w}^{(t)}$ only based on the responsibilities from the current E-step at time t and the previous weights $\mathbf{w}^{(t-1)}$:

$$\begin{aligned} w_j^{(t)} &= \frac{1}{t + \varepsilon(N-1)} \sum_{t'=1}^{t} \gamma_{j,t'} \\ &= \frac{1}{t + \varepsilon(N-1)} \left(\gamma_{j,t} + \sum_{t'=1}^{t-1} \gamma_{j,t'} \right) \\ &= \frac{1}{t + \varepsilon(N-1)} \left(\gamma_{j,t} + (t - 1 + \varepsilon(N-1)) w_j^{(t-1)} \right) \\ &= w_j^{(t-1)} + \frac{1}{t + \varepsilon(N-1)} \left(\gamma_{j,t} - w_j^{(t-1)} \right), \end{aligned} \tag{8}$$

for all $j = 1, ..., N$. The learning rule in Eq. 8 can be written as a delta learning rule as described in the

main text:

$$w_j^{(t)} = w_j^{(t-1)} + \alpha^{(t)} \delta_j^{(t)}, \tag{9}$$

with time-dependent learning rate

$$\alpha^{(t)} = \frac{1}{t + \varepsilon(N-1)} \tag{10}$$

and error

$$\delta_j^{(t)} = \gamma_{j,t} - w_j^{(t-1)} = \left(\frac{k_j(s_t)}{p^{(t-1)}(s_t)} - 1 \right) w_j^{(t-1)}. \tag{11}$$

Note that our learning rule in Eq. 9 is *multiplicative* since the error (Eq. 11) depends on the previous weights $w_j^{(t-1)}$ through a multiplication factor. In that sense, our learning rule is similar to a range of mechanistic models that use multiplicative learning rules for novelty detection ([29, 39, 42, 49–51], see Discussion). The multiplication factor of the weights in Eq. 8 depends on the ratio of kernel activation and familiarity of the current observation s_t, which are both available locally in time. For example, the kernel activation by the current observation s_t can be interpreted as the presynaptic activity of the novelty detection weights $w_j^{(t-1)}$, while the familiarity of s_t can be seen as postsynaptic activity of the novelty-detecting circuit (see Section 1.4 in the SI for a biologically plausible circuit implementation).

Toy experiment: Gabors of different orientations

Novelty predictions of two count-based and two kernel-based novelty models were computed for three sequences of three Gabor stimuli each. Gabor stimuli are characterized by their angular orientation. Across all sequences $i = 1,2,3$, the first stimulus ($s_1^{(i)} == 45°$) and the third stimulus ($s_3^{(i)} = 145°$) are identical. The second stimulus in each sequences shows varying levels of similarity to the first stimulus: $s_2^{(1)}$=45°(identical to $s_1^{(1)}$), $s_2^{(2)} = 60°$ (15° difference to $s_1^{(2)}$), $s_2^{(3)} = 75°$ (30° difference to $s_1^{(3)}$)
Countable states for the count-based novelty models are obtained by binning the space of Gabor orientations $[0, 180°]$ into 90 bins of width 1° (model variant 1) or into four bins of 45° (model variant 2). The size of the state space, $|S|$, is equal to the respective number of bins in each model variant; counts at $t = 0$ were initialized at zero; the prior ε was set to 1.
State representations for the kernel-based novelty models are defined on the 1D torus of Gabor orientations $[0, 180°]$, by defining four equidistantly placed kernels k_j, $j = 1, ..., 4$, centered at $c_1 = 0°$, $c_2 = 45°$, $c_3 = 90°$ and $c_4 = 135°$. The kernels are either triangle-shaped, with kernel function

$$k_j(s) = \frac{1}{\sigma_j} \left(1 - \frac{|s - c_j|}{\sigma_j} \right) \Theta \left(1 - \frac{|s - c_j|}{\sigma_j} \right), \tag{12}$$

or Gaussian-shaped, with kernel function

$$k_j(s) = \frac{1}{\sigma_j \sqrt{2\pi}} \exp \left(-\frac{(s - c_j)^2}{2\sigma_j^2} \right), \tag{13}$$

where $s \in [0°, 180°]$ is the orientation of the Gabor stimulus, c_j as defined above are the centers of the kernels, and σ_j are the 'width' parameters. We choose $\sigma_j = 90°$ for all triangle kernels and $\sigma_j = 72°$ for all Gaussian kernels, creating the overlapping kernels depicted in Fig. 1 B3, B4. Kernel mixture weights at $t = 0$ were initialized uniformly to $\frac{1}{N}$, where $N = 4$ is the number of kernels; the prior ε was set to one.

Passive viewing task: Experimental data by Homann et al.

Setup and recordings. Homann et al. [39] used two-photon calcium imaging to record the neural activity of layer 2/3 neurons in the primary visual cortex (V1) of 5 GCaMP6f-expressing mice during passive viewing of 'familiar' and 'novel' visual stimuli. During the recording, mice were head-fixed but could run freely on an air suspended Styrofoam ball in front of a toroidal screen (field of view in x-direction: $-130°$ to $130°$; field of view in y-direction: $-20°$ to $90°$), onto which the visual stimuli were projected. Homann et al. extracted the peak neural responses (trial- and population-averaged, see SI for details) to the novel image (ΔN) and to the familiar sequence (steady-state activity N_∞) during the 'variable repetition experiment' and the 'variable image number experiment', as well as the transient population response (ΔN_{recov}) to the formerly familiar sequence after the recovery interval in the 'repeated image set experiment'.

Task. They consider three variations of the passive viewing experiment. In the first two variations of the experiment (the 'variable repetition experiment' and the 'variable image number experiment'), a sequence of M 'familiar' stimuli is presented for L repetitions. In the $L+1$-st repetition of the familiar sequence, the last image is replace by a 'novel' image. The familiar sequence is then shown for two additional repetitions, before the next run of the experiment starts with a new set of images for the sequence of familiar images and the novel image. In the 'variable repetition experiment', the number of repetitions L of the familiar sequence is varied ($L = 1,3,8,18,38$, $M = 3$ fixed); in the 'variable image number experiment', the number of visual stimuli M in the familiar is varied ($M = 3,6,9,12$, $L = 18$ fixed). In the third variation of the passive viewing experiment ('repeated image set experiment'), a familiar sequence with M images is presented L times until the neural responses have converged to baseline ($M = 3$, $L = 22$). Then, a different sequence of familiar images is presented for L' repetitions (corresponding to a 'recovery interval' ΔT), before the original sequence is shown again. Neural responses to the formerly familiar sequence are measured for different length of the recovery interval ($\Delta T = 0,21,42,63,84,108,144$ seconds, corresponding to $L' = 0,23,46,70,93,120,160$).

Stimuli. Visual stimuli across all variations of the experiment are shown for 300 seconds each, without blank frames in between. 'Familiar' and 'novel' visual stimuli are drawn from the same distribution of images, each image consisting of a linear superposition of Gabor filters with randomly chosen parameters (see Ref. [39] or next section for details). A 'novel' image is an image that has not previously been observed in the experiment, as opposed to the sequence of 'familiar' images, whose stimuli are 'novel' upon first presentation of the sequence but become increasingly familiar as the sequence is shown repeatedly during a single run of the experiment.

Passive viewing task: Model simulations and fitting

Stimuli. For the simulations, we replace the original images used in the mouse experiment by a simplified stimulus: a single Gabor filter (100% contrast) with fixed location and random angular orientation α between 0 and 180 degrees. The state representations that encode the visual stimuli for the different novelty models are thus directly defined on the interval $[0°, 180°]$ of available Gabor filter orientations. To adhere to the experimental protocol of Homann et al., we need to make sure that all images within a given novel sequence are (i) perceptually different from each other, and (ii) different from the novel image. To this end, we sample the Gabor filter orientations for all images in one run of a given experiment variation as follows: We segment the space of available orientations into N boxes B_i, $i = 1,...,N$ of equal width $|B|$ (denote the lower and upper bounds of each box by l_i and u_i). For each run of the experiment, we then choose an intercept $\Delta\alpha \in [0, |B|)$ uniformly at random between 0 and the box width, and construct the set S_α of all orientations available for that run of the experiment as $S_\alpha = \{\alpha_1, ..., \alpha_N\}$,

where

$$\alpha_i = l_i + \Delta\alpha \quad \in B_i. \tag{14}$$

The orientations for the sequence of familiar images and the novel image are then chosen uniformly at random from the set S_α of available orientations.

Simulation protocol. Three different novelty models were simulated for the passive viewing task: (i) count-based novelty (separate counts for each image), (ii) kernel-based novelty with box-kernels, and (iii) kernel-based novelty with triangle kernels. Model simulations followed the same protocol as the mouse experiment by Homann et al (see previous section). Each simulation step corresponds to 300 ms in the experiment, i.e. in each simulation step, a single stimulus is presented.

Kernel definition. The state representations that encode the visual stimuli during model simulations are defined over the interval $[0°, 180°]$ of possible angular orientations of the presented Gabor filters. For the count-based novelty model, we divide the interval of possible orientations into bins of $1°$ width and define 180 states that each encode one of the bins. For example, state $s = 90$ encodes all orientations between $89°$ and $90°$. For the kernel-based novelty models, we define state representations that consist of K kernel functions k_j over the space $S = [0°, 180°]$ of possible angular orientations s. We consider box-shaped kernels $k_j^\square$ and triangular kernels $k_j^\triangle$, $j = 1, ..., K$, which are respectively characterized by their centers c_j and their width parameter w_j. The box-shaped kernels are defined as

$$k_j^\square(s) = \begin{cases} \frac{1}{w_j} & \text{if } s \in [c_j - \frac{w_j}{2}, c_j + \frac{w_j}{2}], \\ 0 & \text{otherwise.} \end{cases} \tag{15}$$

$$k_j^\square(s) = \frac{K}{180}\left[\Theta\left(c_j - s + w_j\right) - \Theta\left(c_j - s - w_j\right)\right], \tag{16}$$

where s is the angular orientation of the stimulus, and w_j and c_j are the width and center of kernel $k_j^\square$. We choose equally distributed, equal-width box kernels with $w_j = w = \frac{180}{K}$ and centers $c_j = \frac{w}{2} + (j-1)w$ for $j = 1, ..., K$. The triangle-shaped kernels are defined as

$$k_j(s) = \begin{cases} 1 - \frac{|s - c_j|}{w_j} & \text{if } s \in [c_j - w_j, c_j + w_j], \\ 0 & \text{otherwise,} \end{cases} \tag{17}$$

$$k_j(s) = \left(1 - \frac{|s - c_j|}{w_j}\right)\Theta\left(1 - \frac{|s - c_j|}{w_j}\right), \tag{18}$$

where s is the angular orientation of the stimulus, w_j is the width parameter of kernel k_j (we choose $w_j = ...$ which yields overlapping triangles as depicted in Fig. 1 B3, Fig. 2 C), and the kernel centers c_j are given as We choose equally distributed, equal-width triangular kernels with $w_j = w = \frac{180}{2K}$ and centers $c_j = (j-1)w$ for $j = 1, ..., K$, which yields overlapping triangle kernels as depicted in Fig. 2 C. The number of kernels N, the width $\sigma_j = \sigma$ (shared across all kernels) and the prior ε that determines the learning rate of the kernel-based novelty update are free parameters that are fitted to neural data for each model (see next section).

Fit to neural data. To fit each novelty model to the neural data, we use grid search in combination with linear regression: we simulate a given model $\mathscr{M}$ across the full range of model parameters $\theta = (N, \sigma, \varepsilon)$ (see SI, Section **??**); we then fit the simulated novelty responses to the neural data using linear regression (see SI, Section 3.1). The parameter set that minimizes this regression error is chosen as the best-fit parameter set for model $\mathscr{M}$. We cross-validate our fitting results by fitting model $\mathscr{M}$ to the neural data from two out of three experiments and computing the cross-validation error as the regression score (MSE/variance) for the fitted model on the left-out data set (see SI, Section 3.1).

Active exploration task: Experimental data by Rosenberg et al.

Task. Rosenberg et al. [52] track the behaviour of 20 mice (10 of them water-deprived) while they successively had free access to an unfamiliar maze (maze cleaned with ethanol between mice). Each mouse was video-recorded during its first 7 hours of access to the maze, during the dark portion of the animal's light cycle. The maze (dimensions: 24×24×2 inches, corridors: 1-1/8 inches wide, ≈ 12-1.5 inches long) was constructed as a 6-level binary tree, with branching points and end points of corridors as nodes. A single end node in the maze contained a water delivery port, which delivered a water reward to water-deprived mice ('rewarded group') but was inactive during the experiment with non-deprived mice ('unrewarded group'). The labyrinth walls and floor were opaque for the visual spectrum of the mice but permeable for infrared light to allow tracking of mouse behaviour using infrared cameras. The x-y-coordinates of the animal's nose in each video frame, extracted with a version of DeepLabCut, were used as the 'mouse position' for subsequent behavioral analysis (both by Rosenberg et al. and in our study).

Environment formalization. To enable direct comparison between the behavior of mice and reinforcement learning (RL) models, we formalized the labyrinth environment using 'states' and 'actions', where 'states' s characterize the possible locations of agents (mice or RL models) in the labyrinth maze, and 'actions' a characterize the possible next direction the agent can take from a given state. Analogously to Rosenberg et al. [52], the home cage and all branching points and end points of the labyrinths corridors (i.e. all nodes in the binary tree) were defined as 'states'. There are four possible actions available in a given state in the maze, depending on the structure of the labyrinth: (i) go forward into maze (available in the home cage); (ii) go back out of the maze (available in all states except the home cage); (iii) go into the left corridor (available in all branching points) and (iv) go into the right corridor (available in all branching points). The notion of 'left' and 'right' is based on an allocentric perspective, i.e. relative to the position of the home cage (not relative to the moving direction of the mouse). Based on the definition of states in the maze, mouse behavior over time is discretized into 'time points', where each time point marks the transition to a new state in the maze.

Preprocessing. For each mouse in Rosenberg et al.'s data set (github: [70]), we extract the sequence of states and actions taken at each time point between their first entry into the maze and their first encounter with the goal state. To exclude any reward-related behaviors from the analysis, we disregard any mouse behavior after the first encounter with the goal state, independently of whether the mouse received a reward in the goal state (rewarded group) or not (unrewarded group). Since there are no significant behavioral differences between rewarded and unrewarded mice during the time before the first reward encounter [52], we treat them as equivalent (see SI, Section 6 for details on outliers).

Novelty-seeking reinforcement learning (N-RL) models

We adapt classical reinforcement learning (RL) algorithms to model novelty-seeking ('novelty-seeking RL (N-RL) agents') by replacing the extrinsic reward signal by an intrinsically computed novelty signal (see e.g. [15, 18, 25, 32, 53] for similar approaches).

Like classical RL agents, N-RL agents move in an environment which is defined by states s and actions a, which allow the agent to transition between states. States usually represent the spatial location of the agent, but can also include other sensory inputs or internal state variables. The environment can be probabilistic in the sense that a given action a in a given state s can lead the agent into different states s'. The probabilities for each such state transition (s, a, s') are called transition probabilities and summarized in the matrix $p \in \mathbb{R}^{|S|\times|A|\times|S|}$, where $|S|$ is the size of the state space S, and $|A|$ is the size of the action space A.

The main difference to classical RL algorithms is that, instead of maximizing the return of extrinsic rewards, the N-RL algorithms maximize the novelty return:

$$G_{N,t} = \sum_{t'=t}^{T_{\max}} \gamma^{t'} N^{(t')}(s_{t'}), \tag{19}$$

where the novelty $N^{(t')}(s_{t'})$ is computed using any suitable novelty model, e.g. count-based or kernel-based novelty. Analogous to reward-based V-values and Q-values, N-RL algorithms compute 'novelty-based' V-values and Q-values:

$$V_N^{(t)}(s) = \mathbb{E}\left[\sum_{t'=1}^{\infty} \gamma^{t'} N^{(t')}(s_{t'}) | s_1 = s\right], \tag{20}$$

$$Q_N^{(t)}(s,a) = \mathbb{E}\left[\sum_{t'=1}^{\infty} \gamma^{t'} N^{(t')}(s_{t'}) | s_1 = s, a_1 = a\right]. \tag{21}$$

Active exploration task: Model simulations and fitting

We simulate and fit a total of 24 novelty-seeking RL (N-RL) models to the exploration behavior of the mice in the Rosenberg maze, and compare the fitted models with respect to their log-evidence. We consider models from three different RL model families: (i) model-based N-RL models based on Prioritized Sweeping [71] (Alg. 1 in SI), (ii) model-free N-RL models with Actor-Critic architecture [71] (Alg. 3 in SI), and (iii) hybrid N-RL models that combines the model-free and model-based variants using a hybrid action policy (Alg. 4 in SI). For each N-RL model family, we consider one model with count-based novelty, and seven models with kernel-based novelty, one for each of the kernel-based representations (see below).

State representations for count-based and kernel-based novelty. Count-based novelty in the Rosenberg maze uses a separate count for each state as defined above, i.e. each node in the binary tree representation of the maze. Using Eqs. 1 and 2, the novelty reward at each time is then computed as the novelty of the current state in the environment. We define kernel-based novelty in the Rosenberg maze with respect to seven different kernel-based state representations that differ in the degree of granularity with which they encode the maze.

The kernel representations are based on the idea that each kernel should encode a different 'area' of the labyrinth environment, e.g. the left or right half of the maze (allocentric perspective), as well as a trace of how to reach this 'area' from the home cage. Depending on the level of 'granularity' of the state representation, the kernels each encode larger areas (low granularity) or smaller areas (high granularity). To formally write the corresponding kernel functions, we consider the binary-tree structure of the labyrinth maze (Fig. 4C2) and define the '(sub-)tree' $\text{Tree}(s)$ for any state s in the labyrinth ($s \in S$) as

$$\text{Tree}(s) = \{s' \in S : s' \text{ is part of a binary tree with root state } s\}. \tag{22}$$

Hence, the entire labyrinth (excluding the home state) can be written as $\text{Tree}(s_0)$ where s_0 is the first branching point of the labyrinth, whereas the sub-tree of each leaf node is the leaf node itself. We further define the sets S_ℓ, $\ell = \{0, 1, ..., 6\}$, where each set S_ℓ contains all 2^ℓ states in a given level ℓ of the binary tree. For example, the set S_0 contains only s_0, the first branching point of the maze, while S_6 contains all 64 leaf nodes.

We define the kernel representation of granularity level ℓ, for $\ell = 1, ..., 6$ as follows. The kernel representation of a given granularity ℓ consists of $2^\ell + 1$ kernels: first, the home cage kernel, that is the same

across all granularity levels and encodes the home cage as a separate state:

$$k_H(s) = \begin{cases} 1 & \text{if } s = s_H \\ 0 & \text{otherwise;} \end{cases} \tag{23}$$

and second, the kernels $k_j^{(\ell)}$ that each encode the 'area' around one of the states $s_j^{(\ell)} \in S_\ell = \{s_1^{(\ell)}, ..., s_{2^\ell}^{(\ell)}\}$:

$$k_j^{(\ell)} = \begin{cases} \frac{1}{K^{(\ell)}} & \text{if } s \in \text{Tree}(s_j^{(\ell)}) & \text{('area')} \\ \frac{1}{2^{\ell-\ell'}K^{(\ell)}} & \text{if } s \in S_{\ell-\ell'} \text{ and } s_j^{(\ell)} \in \text{Tree}(s),\ \ell' = 0, ..., \ell & \text{('trace')} \\ 0 & \text{otherwise.} \end{cases} \tag{24}$$

The factor $K^{(\ell)} = K_j^{(\ell)} = \sum_{s \in S} k_j^{(\ell)}(s)$ is added to normalize the kernels to 1 over the state space.

Model fitting. We fit the parameters of each N-RL model to mouse behavior using maximum-likelihood estimation (MLE):

$$\hat{\theta} = \arg\max_\theta \text{LL}(\mathscr{D}, \mathscr{M}, \theta) = \arg\max_\theta (\log P(\mathscr{D}|\mathscr{M}, \theta)), \tag{25}$$

where $\mathscr{M}$ is the N-RL model that is being fitted, θ are its model parameters, and $\mathscr{D}$ is the mouse data. The mouse data $\mathscr{D}$ consists of the appended state-actions sequences of all mice during their exploration of the labyrinth:

$$\mathscr{D} = (s_0^{(1)}, a_0^{(1)}, ..., a_{T_1-1}^{(1)}, s_{T_1}^{(1)}, \quad ... \quad, s_0^{(20)}, a_0^{(20)}, ..., a_{T_{20}-1}^{(20)}, s_{T_{20}}^{(20)}), \tag{26}$$

where $s_t^{(i)}$ and $a_t^{(i)}$ denote the state and action of mouse i at time point t, and T_i denotes the number of exploration steps of mouse i until its first encounter of the goal state.

We maximize the log-likelihood for a given model by minimizing its negative log-likelihood using the scipy.optimize implementation of the Nelder-Mead algorithm (best performing among relevant optimization algorithms, including L-BFGS-B, SLSQP). In each minimization step, the log-likelihood for the current parameter set θ is computed (see SI, Section 5), and its negative is used as input to the minimizer of the scipy.optimize package to compute the next candidate parameter set. We run the optimization until convergence to obtain the fitted parameters for a given model.

Bayesian model comparison. We compare fitted models with respect to their their log-evidence (LE):

$$\text{LE}(\mathscr{D}, \mathscr{M}) = \text{LL}(\mathscr{D}, \mathscr{M}, \theta)) - \frac{|\theta_{\mathscr{M}}|}{2} \log(T), \tag{27}$$

where the log-likelihood term accounts for the fit of the model, and the remaining term introduces a cost for model complexity, that is a function of the number of free parameters $|\theta_{\mathscr{M}}|$ of the model, and the number of data points $T = \sum_{i=1}^{20} T_i$ in the fitted data set $\mathscr{D}$.

Posterior predictive checks. We perform posterior predictive checks to verify that the winning model from our model comparison matches the fitted data (i.e. mouse behavior) with respect to relevant statistics of their behavior. To this end, we simulated all fitted models ($n = 500$ instantiations for each model). Each model simulation starts with the model agent being initialized in the home state, and ends when the model agent first reaches the goal state.

We compare mice and simulated models with respect to (i) the average number of steps in the maze until reaching the goal state; (ii) the average number of novel end nodes visited relative to the total number of visits to a (familiar or novel) end node; (iii) the average efficiency coefficient N_{32}, i.e. the total number of end node visits until the agent 32 (i.e. half) of the end nodes in the maze have been discovered; (iv) the

integrated integrated difference between the average end nodes discovery curve of mice and the average end node discovery curve of a given model. Error bars indicated the standard error of the mean (sem). For the fourth statistic, the sem is computed using bootstrapping on the set of end node discovery curves of individual agents of a given type.

Supplementary Information

1 Derivation of EM updates

As described in the main manuscript, estimate the familiarity density $p^{(t)}$ at time point t,

$$p^{(t)}(s) = \sum_{j=1}^{N} w_j^{(t)} k_j(s), \tag{S.1}$$

by choosing the kernel mixture weights at time t, $\mathbf{w}^{(t)} = \left(w_1^{(t)}, ..., w_N^{(t)}\right)^{\mathrm{T}}$, as the maximum-a-posteriori (MAP) estimate of the sequence $s_{1:t}$ of observations up to t:

$$\mathbf{w}^{(t)} = \arg\max_{\mathbf{w}} \log \Pr(s_{1:t}|\mathbf{w}) \Pr(\mathbf{w}), \tag{S.2}$$

where $\Pr(\mathbf{w})$ is a Dirichlet prior over the weights, i.e.

$$\Pr(\mathbf{w}) = c(\alpha) \prod_{j=1}^{N} (w_j)^{\alpha_j - 1}, \tag{S.3}$$

with concentration parameters $\alpha = (\alpha_1, ..., \alpha_N)^T$ and

$$c(\alpha) = \frac{\Gamma\left(\sum_{j=1}^{N} \alpha_j\right)}{\prod_{j=1}^{N} \Gamma\left(\alpha_j\right)}. \tag{S.4}$$

To compute the MAP estimate of the weights, we use the EM algorithm which allows us to compute the MAP estimate in an iterative fashion. The underlying idea is that, by introducing latent variables, the loglikelihood in Eq. S.2 can be decomposed into two terms that can be iteratively optimized, yielding the so-called E-step and M-step of the EM algorithm. We first derive the full EM updates for our model setup. We then show how these updates can be simplified in the incremental EM setup, a variant of EM that is more appropriate to our setting (see below for a discussion of our choice of EM algorithm).

1.1 Classical EM for our framework

Latent variables. To apply the EM algorithm to our kernel-mixture model, we define vectors of binary latent random variables $\mathbf{z}_{t'} = \left(z_{1,t'}, ..., z_{N,t'}\right)^{\mathrm{T}}$ for each time step $t' = 1, ..., t$, such that the following three conditions hold:

$$\Pr\left(z_{j,t'} = 1|\mathbf{w}\right) = w_j \qquad \text{for all } j = 1, ..., N \text{ and } t' = 1, ..., t \tag{S.5}$$

$$\Pr\left(s_{t'}|z_{j,t'} = 1\right) = k_j(s_{t'}) \qquad \text{for all } j = 1, ..., N \text{ and } t' = 1, ..., t \tag{S.6}$$

$$\sum_{j=1}^{N} z_{j,t'} = 1 \qquad \text{for all } t' = 1, ..., t. \tag{S.7}$$

Note that the last condition, Eq. S.7, implies that each vector $\mathbf{z}_{t'}$ is a one-hot-coding vector of the N kernel indices. The probability of a given $\mathbf{z}_{t'}$ to be a one-hot-coding vector of index j is given by the kernel weight w_j (Eq. S.5), while the probability of observing stimulus $s_{t'}$ when $\mathbf{z}_{t'}$ is encoding index j is

the value of kernel j at stimulus $s_{t'}$ (Eq. S.6). Effectively, the latent variables thus link the kernel values for a given stimulus and the kernel weights (add SI figure).

Eq. S.5 and Eq. S.6 allow us to write conditional distributions for the stimulus and latent variable at time t', $s_{t'}$ and $\mathbf{z}_{t'}$:

$$p(s_{t'}|\mathbf{z}_{t'}) = \prod_{j=1}^{N} \left(k_j(s_{t'})\right)^{z_{j,t'}} \tag{S.8}$$

$$p(\mathbf{z}_{t'}|w) = \prod_{j=1}^{N} (w_j)^{z_{j,t'}}. \tag{S.9}$$

We consider the observations $s_{t'}$ to be independent (see discussion below). That allows us to further write the joint distributions

$$p(s_{1:t}|\mathbf{z}_{1:t}) = \prod_{t'=1}^{t} P(s_{t'}|\mathbf{z}_{t'}) = \prod_{t'=1}^{t}\prod_{j=1}^{N} \left(k_j(s_{t'})\right)^{z_{j,t'}} \tag{S.10}$$

$$p(\mathbf{z}_{1:t}|w) = \prod_{t'=1}^{t} P(\mathbf{z}_{t'}|w) = \prod_{t'=1}^{t}\prod_{j=1}^{N} (w_j)^{z_{j,t'}}. \tag{S.11}$$

These distributions will give us the explicit expressions for the E-step and M-step that we will now derive via the loglikelihood decomposition.

Loglikelihood decomposition. The latent variables $\mathbf{z}_t$ allow for a decomposition of the loglikelihood in Eq. 5. Let $q(\mathbf{z}_{1:t})$ be a arbitrary distribution over the latent variables $\mathbf{z}_{1:t}$, then we can write the loglikelihood as

$$\log \Pr(s_{1:t}|\mathbf{w}) \Pr(\mathbf{w}) \overset{\text{law of total prob.}}{=} \sum_{\mathbf{z}_{1:t}} q(\mathbf{z}_{1:t}|\mathbf{w}) \log \Pr(s_{1:t}|\mathbf{w}) + \log \Pr(\mathbf{w}) \tag{S.12}$$

$$= \sum_{\mathbf{z}_{1:t}} q(\mathbf{z}_{1:t}|\mathbf{w}) \log \left(\frac{\Pr(s_{1:t}, \mathbf{z}_{1:t}|\mathbf{w})}{\Pr(\mathbf{z}_{1:t}|s_{1:t}, \mathbf{w})}\right) + \log \Pr(\mathbf{w}) \tag{S.13}$$

$$= \sum_{\mathbf{z}_{1:t}} q(\mathbf{z}_{1:t}|\mathbf{w}) \log \left(\frac{\Pr(s_{1:t}, \mathbf{z}_{1:t}|\mathbf{w})}{q(\mathbf{z}_{1:t}|\mathbf{w})}\right) + \log \Pr(\mathbf{w}) - \sum_{\mathbf{z}_{1:t}} q(\mathbf{z}_{1:t}|\mathbf{w}) \log \left(\frac{\Pr(\mathbf{z}_{1:t}|s_{1:t}, \mathbf{w})}{q(\mathbf{z}_{1:t}|\mathbf{w})}\right) \tag{S.14}$$

$$= \mathscr{L}(q(\mathbf{z}_{1:t}), p(s_{1:t}, \mathbf{z}_{1:t})) + \mathrm{KL}(q(\mathbf{z}_{1:t})||p(\mathbf{z}_{1:t}|s_{1:t})), \tag{S.15}$$

where $p(\mathbf{z}_{1:t}|s_{1:t}) = \Pr(\mathbf{z}_{1:t}|s_{1:t}, \mathbf{w})$. Based on this decomposition, the loglikelihood can be maximized by iteratively (i) minimizing the KL divergence (E-step) while keeping the weights $\mathbf{w}$ fixed at their prior values, and (ii) maximizing the Q term with respect to $\mathbf{w}$ while keeping the latent distribution q fixed (M-step). For the proof that this iterative procedure indeed maximizes the loglikelihood, see e.g. Ref. [48, 69].

E-step. The KL-divergence of q and p becomes minimal when the two distributions are equal, so in the

E-step, we set

$$q(\mathbf{z}_{1:t}|\mathbf{w}) = p(\mathbf{z}_{1:t}|s_{1:t},\mathbf{w}) \tag{S.16}$$

$$= \frac{\Pr(s_{1:t}|\mathbf{z}_{1:t},\mathbf{w}^{(t)})\Pr(\mathbf{z}_{1:t}|\mathbf{w})}{\Pr(s_{1:t}|\mathbf{w})} \tag{S.17}$$

$$= \prod_{t'=1}^{t}\prod_{j=1}^{N}\left(\frac{w_j k_j(s_{t'})}{\sum_{j'=1}^{N} w_{j'} k_{j'}(s_{t'})}\right)^{z_{j,t'}} \tag{S.18}$$

$$= \prod_{t'=1}^{t}\prod_{j=1}^{N}\left(\gamma_{j,t'}\right)^{z_{j,t'}}, \tag{S.19}$$

where we have used the mutual independence of any two observations $s_{t'}$, $s_{t''}$ and any two latent variables $\mathbf{z}_{t'}$ and $\mathbf{z}_{t''}$, respectively to obtain Eq. S.17, and substituted the distributions from Eq. 3, Eq. S.8 and Eq. S.9 to obtain Eq. S.18. To compute the distribution q in the E-step, it is thus sufficient to compute the terms

$$\gamma_{j,t'} = \frac{w_j k_j(s_{t'})}{\sum_{j'=1}^{N} w_{j'} k_{j'}(s_{t'})}, \tag{S.20}$$

also called 'responsibilities', for every $j = 1,\ldots,N$ and every observed stimulus $s_{t'}$, $t' = 1,\ldots,t$.

M-step. To maximize the term $\mathscr{L}(q(\mathbf{z}_{1:t}), p(s_{1:t},\mathbf{z}_{1:t}))$ with respect to $\mathbf{w}$, we keep the latent distribution q fixed (denoted by fixed weights $\mathbf{w}_{\text{E-step}}$ that parametrize q). Note that

$$\mathscr{L}(q(\mathbf{z}_{1:t}), p(s_{1:t},\mathbf{z}_{1:t})) = \sum_{\mathbf{z}_{1:t}} q(\mathbf{z}_{1:t}|\mathbf{w}_{\text{E-step}}) \log\left(\frac{p(s_{1:t},\mathbf{z}_{1:t}|\mathbf{w})}{q(\mathbf{z}_{1:t}|\mathbf{w}_{\text{E-step}})}\right) + \log\Pr(\mathbf{w}) \tag{S.21}$$

$$= \sum_{\mathbf{z}_{1:t}} q(\mathbf{z}_{1:t}|\mathbf{w}_{\text{E-step}}) \log p(s_{1:t},\mathbf{z}_{1:t}|\mathbf{w}) + \log\Pr(\mathbf{w}) - \sum_{\mathbf{z}_{1:t}} q(\mathbf{z}_{1:t}|\mathbf{w}_{\text{E-step}}) \log q(\mathbf{z}_{1:t}|\mathbf{w}_{\text{E-step}}). \tag{S.22}$$

Since we keep the latent distribution q fixed throughout the maximization, the last term in Eq. S.22 is constant in $\mathbf{w}$. It is thus sufficient to maximize

$$Q(\mathbf{w},\mathbf{w}^{(t)}) = \sum_{\mathbf{z}_{1:t}} q(\mathbf{z}_{1:t}|\mathbf{w}) \log p(s_{1:t},\mathbf{z}_{1:t}|\mathbf{w}) + \log\Pr(\mathbf{w}) \tag{S.23}$$

$$= \sum_{\mathbf{z}_{1:t}} q(\mathbf{z}_{1:t}|\mathbf{w}) \log\Pr(s_{1:t}|\mathbf{z}_{1:t},\mathbf{w})\Pr(\mathbf{z}_{1:t}|\mathbf{w}) + \log\Pr(\mathbf{w}) \tag{S.24}$$

$$= \sum_{\mathbf{z}_{1:t}} \prod_{t'=1}^{t}\prod_{j=1}^{N}\left(\gamma_{j,t'}\right)^{z_{j,t'}} \log\left(\prod_{t'=1}^{t}\prod_{j=1}^{N}\left(w_j k_j(s_{t'})\right)^{z_{j,t'}}\right) + \log\Pr(\mathbf{w}) \tag{S.25}$$

$$= \sum_{\mathbf{z}_{1:t}} \prod_{t'=1}^{t}\prod_{j=1}^{N}\left(\gamma_{j,t'}\right)^{z_{j,t'}} \sum_{t'=1}^{t}\sum_{j=1}^{N} z_{j,t'} \log\left(w_j k_j(s_{t'})\right) + \log\Pr(\mathbf{w}), \tag{S.26}$$

where we have substituted the distributions in Eq. S.8, Eq. S.9 and the expression for q from the E-step. Since all $\mathbf{z}_{t'}$ are one-hot coding vectors, exactly one $z_{t',j}$ is non-zero for each t'; and since we are summing over all possible one-hot coding sequences of vectors $\mathbf{z}_{1:t}$, each $z_{j,t'}$ will be non-zero exactly once. Eq. S.26 therefore reduces to

$$Q(\mathbf{w},\mathbf{w}^{(t)}) = \sum_{t'=1}^{t}\sum_{j=1}^{N} \gamma_{j,t}\left(\log k_j(s_{t'}) + \log w_j\right) + \log\Pr(\mathbf{w}). \tag{S.27}$$

We maximize Q as in Eq. S.27 under the constraint that all weights sum to one using Lagrange multipli-

ers, by solving

$$0 = \frac{\mathrm{d}}{\mathrm{d}w_j}\left(Q(\mathbf{w},\mathbf{w}^{(t)}) - \lambda\left(\sum_{j'=1}^{N} w_{j'} - 1\right)\right) = \frac{1}{w_j}\sum_{t'=1}^{t}\gamma_{j,t'} + \frac{\mathrm{d}}{\mathrm{d}w_j}\log\Pr(\mathbf{w}) - \lambda \tag{S.28}$$

for all $j = 1,...,N$. Since the Lagrange multiplier has to satisfy Eq. S.28 for all j, it has to satisfy the following equation (after multiplying Eq. S.28 with w_j and summing over j):

$$\begin{aligned} 0 &= \sum_{j=1}^{N}\sum_{t'=1}^{t}\gamma_{j,t'} + \sum_{j=1}^{N} w_j \frac{\mathrm{d}}{\mathrm{d}w_j}\log\Pr(\mathbf{w}) - \lambda\sum_{j=1}^{N} w_j && \text{(S.29)}\\ &= \sum_{t'=1}^{t}\sum_{j=1}^{N}\gamma_{j,t'} + \sum_{j=1}^{N} w_j \frac{\mathrm{d}}{\mathrm{d}w_j}\log\Pr(\mathbf{w}) - \lambda\sum_{j=1}^{N} w_j && \text{(S.30)}\\ &\overset{\text{Eq. S.7}}{=} t + \sum_{j=1}^{N} w_j \frac{\mathrm{d}}{\mathrm{d}w_j}\Pr(\mathbf{w}) - \lambda, && \text{(S.31)} \end{aligned}$$

yielding

$$\lambda = t + \sum_{j=1}^{N} w_j \frac{\mathrm{d}}{\mathrm{d}w_j}\log\Pr(\mathbf{w}). \tag{S.32}$$

Substitution in Eq. S.28 gives the solution for the new kernel mixture weights:

$$w_j = \frac{1}{t + c_j(w)}\sum_{t'=1}^{t}\gamma_{j,t'}, \tag{S.33}$$

where

$$c_j(w) = \sum_{j'\neq j} w_{j'}\frac{\mathrm{d}}{\mathrm{d}w_{j'}}\log\Pr(\mathbf{w}) \tag{S.34}$$

Special Dirichlet priors. For our Dirichlet prior (Eq. S.3), $c_j(w)$ reduces to

$$\begin{aligned} c_j(w) &= \sum_{j'\neq j} w_{j'}\frac{\mathrm{d}}{\mathrm{d}w_{j'}}\log\Pr(\mathbf{w}) && \text{(S.35)}\\ &= \sum_{j'\neq j} w_{j'}\frac{\mathrm{d}}{\mathrm{d}w_{j'}}\left(\log c(\alpha) + \sum_{j''=1}^{N}(\alpha_{j''}-1)\log w_{j''}\right) && \text{(S.36)}\\ &= \sum_{j'\neq j}(\alpha_{j'}-1). && \text{(S.37)}\\ & && \text{(S.38)} \end{aligned}$$

For Dirichlet prior with shared concentration parameter $\alpha_j = \varepsilon + 1$ for all $j = 1,...,N$, $c_j(w)$ reduces to $(N-1)\varepsilon$ for all j. In this case, the M-step update (Eq. S.33) reduces to

$$w_j = \frac{1}{t + (N-1)\varepsilon}\sum_{t'=1}^{t}\gamma_{j,t'}. \tag{S.39}$$

A Dirichlet prior with shared concentration parameter encodes the assumption that all kernels have been observed equally prior to observing the stimulus sequence $s_{1:t}$ – a reasonable assumption that we will make for the rest of the derivation.

1.2 Incremental EM

Full EM. In summary, the full EM algorithm estimates the kernel mixture weights that maximize the likelihood of our observations $s_{1:t}$ in two alternating steps: first, the 'responsibilities' $\gamma_{j,t'}$ are updated for all $j = 1,...,N$ and $t' = 1,...,t$ as in Eq. S.20 (E-step); then the weights $w_j^{(t)}$ after observations $s_{1:t}$ are computed as in Eq. S.39 (M-step). In the full EM setup, these two update steps are repeated until convergence of the algorithm. Since the full EM relies on knowledge about all observations $s_{1:t}$ to make a single update, the mixture weights can only be estimated after observing the full sequence $s_{1:t}$ that is then processed together in every iteration of the EM algorithm ('batch mode').

Incremental EM (I-EM). For our goal of modeling human and animals, however, this 'batch' update of the familiarity density (and hence, of stimulus novelty) is undesired since we assume that the brain updates its estimate of the novelty of a stimulus immediately after its observation – even if that leads to a less precise estimate of the overall statistics of stimuli in the environment. Instead of the full EM algorithm, we therefore estimate our kernel weights using the incremental EM algorithm. This variant of EM that uses only a single observation in each update iteration, iterating through the sequence of observations $s_{1:t}$ one-by-one, and then starting again at the beginning of the sequence until the algorithm has converged.

Continual incremental EM (CI-EM). For our scenario of humans and animals experiencing an unfamiliar environment, we assume that the incremental EM algorithms uses a continuing sequence of observations for its estimates, such that each observation is used only once for the update of the algorithm. The resulting weight estimates are in that sense 'approximate' estimates of the kernel weights but also the best estimates that are available for a potentially infinite sequence of observations. Moreover, in environments with finitely many distinguishable stimuli, all stimuli will eventually be revisited and thus contribute again to the weight estimate (like in the incremental EM for finite observation sequences). An important difference is, thought, that the proportion of how much each stimulus is revisited and thus contributes to the estimate of the familiarity distribution is not determined by a fixed circular schedule but by the experimenter (passive viewing task) or by the human or animal through its action policy (active exploration). To distinguish our specific use of the incremental EM algorithm from its classical 'finite stimulus sequence' usage, we also refer to our algorithm as 'continual incremental EM algorithm'.

Continual incremental EM (CI-EM) updates. The main difference between full EM and incremental EM updates is that the latter only uses a single observation $s_{t'}$ in its E-step update: instead of recomputing the responsibilities for all latent variables $z_{j,t''}$, for all $j = 1,...,N$ and all $t'' = 1,...,t$, the incremental E-step only recomputes the responsibilities for the observation $s_{t'}$; all responsibilities from previous iterations of the algorithm are kept the same:

$$\gamma_{j,t''}^{(i)} = \begin{cases} \frac{w_j^{(i-1)} k_j(s_{t''})}{\Sigma_{j'=1}^{N} w_{j'}^{(i-1)} k_{j'}(s_{t''})} & \text{if } t'' = t' \\ \gamma_{j,t''}^{(i-1)} & \text{else.} \end{cases} \tag{S.40}$$

where the additional index i denotes the ith iteration of the algorithm, in which observation $s_{t'}$ is considered. The M-step of the algorithm is the same as in the full EM algorithm (Eq. S.39) – just the underlying responsibilities are different due to the incremental E-step.

In our continual I-EM algorithm, each observation s_t is followed by one iteration of the I-EM algorithm,

i.e. the t-th step of algorithm considers the t-th observation:

$$\gamma_{j,t'}^{(t)} = \begin{cases} \frac{w_j^{(t-1)} k_j(s_{t'})}{\sum_{j'=1}^{N} w_{j'}^{(t-1)} k_{j'}(s_{t'})} & \text{if } t' = t \\ \gamma_{j,t'}^{(t-1)} & \text{else.} \end{cases} \tag{S.41}$$

In this scenario, the t-th M-step update (Eq. S.39) can be written as

$$w_j^{(t)} = \frac{1}{t + (N-1)\varepsilon} \sum_{t'=1}^{t} \gamma_{j,t'}^{(t)} = \frac{1}{t + (N-1)\varepsilon} \sum_{t'=1}^{t} \gamma_{j,t'}^{(t')}, \tag{S.42}$$

where $w_j^{(t')}$ for $j = 1, ..., N$ are the kernel weights at time t' in the experiment. Eq. S.42 gives rise to an iterative update rule (also see Methods, where we dropped the iteration index t' for better readability):

$$\begin{aligned} w_j^{(t)} &= \frac{1}{t + \varepsilon(N-1)} \sum_{t'=1}^{t} \gamma_{j,t'}^{(t')} \\ &= \frac{1}{t + \varepsilon(N-1)} \left(\gamma_{j,t}^{(t)} + \sum_{t'=1}^{t-1} \gamma_{j,t'}^{(t')} \right) \\ &= \frac{1}{t + \varepsilon(N-1)} \left(\gamma_{j,t}^{(t)} + (t - 1 + \varepsilon(N-1)) w_j^{(t-1)} \right) \\ &= w_j^{(t-1)} + \frac{1}{t + \varepsilon(N-1)} \left(\gamma_{j,t}^{(t)} - w_j^{(t-1)} \right), \end{aligned} \tag{S.43}$$

where $\gamma_{j,t}^{(t)}$ is given by Eq. S.41.

1.3 Count-based novelty as a special case of kernel-based novelty

In a scenario where all observations are (perfectly) distinct, count-based novelty and kernel-based novelty are equivalent. To formalize this statement, consider a sequence of observations $s_{1:t}$, where each observation is drawn out of a set of N perfectly distinct stimuli, $S = \{s_1, ..., S_N\}$. We say that count-based novelty and kernel-based novelty are equivalent with respect to this sequence if, at all time steps $t' = 0, ..., t$ and for all stimuli $s_j \in S$, their familiarity distributions $p_c^{(t')}$ (count-based) and $p_k^{(t')}$ (kernel-based) take identical values, i.e.

$$p_c^{(t')}(s_j) = p_k^{(t')}(s_j) \quad \text{for all } t' = 0, ..., t \text{ and all } j = 1, ..., N. \tag{S.44}$$

We now show that Eq. S.44 holds for count-based and kernel-based novelty. To this end, first note that the count-based familiarity at each time point t is given as

$$p_c^{(t)}(s) = \sum_{j=1}^{N} \delta(s, s_j) \frac{C_t(s_j) + \varepsilon_c}{t + N\varepsilon_c}, \tag{S.45}$$

where ε_c is the count-based prior and δ is the Kronecker delta function. We can rewrite the count-base familiarity distribution (Eq. S.45) in the style of a kernel-based familiarity density (i.e. as a mixture model) as follows:

$$p_c^{(t)}(s) = \sum_{j=1}^{N} \delta(s, s_j) \tilde{w}_j^{(t)}, \tag{S.46}$$

with

$$\tilde{w}_j^{(t)} = \frac{C_t(s_j) + \varepsilon_c}{t + N\varepsilon_c}. \tag{S.47}$$

Since the counts are initialized as $C_0(s_j) = 0$ for all $j = 1, ..., N$, the initial 'count-based' mixture weights at time $t = 0$ are given as

$$\tilde{w}_j^{(0)} = \frac{1}{N}, \tag{S.48}$$

for all stimuli $j = 1, ..., N$. Since counts are updated at subsequent time steps t as

$$C_t(s_j) = C_{t-1}(s_j) + \delta(s_t, s_j), \tag{S.49}$$

the update of the count-based mixture weights is given as:

$$\begin{aligned}
\tilde{w}_j^{(t)} &= \frac{C_t(s_j) + \varepsilon_c}{t + N\varepsilon_c} && \text{(S.50)}\\
&\overset{\text{Eq. S.49}}{=} \frac{C_{t-1}(s_j) + \varepsilon_c}{t + N\varepsilon_c} + \frac{\delta(s_t, s_j)}{t + N\varepsilon_c} && \text{(S.51)}\\
&= \frac{C_{t-1}(s_j) + \varepsilon_c}{t - 1 + N\varepsilon_c} \frac{t - 1 + N\varepsilon}{t + N\varepsilon} + \frac{\delta(s_t, s_j)}{t + N\varepsilon_c} && \text{(S.52)}\\
&= \tilde{w}_j^{(t-1)} \left(1 - \frac{1}{t + N\varepsilon}\right) + \frac{\delta(s_t, s_j)}{t + N\varepsilon_c}. && \text{(S.53)}
\end{aligned}$$

We now construct a kernel-based novelty model that is equivalent to the previous count-based novelty model. Since all observations s_j are perfectly distinct, we can construct a continuous stimulus space $S' \supset S$ and non-overlapping, box-shaped kernels k_j, $j = 1, ..., N$, such that

$$k_j(s) = \begin{cases} 1 & \text{if } s = s_j, \\ 0 & \text{otherwise.} \end{cases} \tag{S.54}$$

and $\int_{S'} k_j(s)\,\mathrm{d}s$ for all $j = 1, ..., N$ (e.g., we can place the s_j on a real line with equal distance 1 between them, and define box kernels of height and width 1, centered at the different s_j). Then the familiarity distribution of the kernel-novelty model at stimulus $s_i \in S$ is given as

$$p_k^{(t)}(s_i) = \sum_{j=1}^{N} \delta(s_i, s_j) w_j^{(t)}, \tag{S.55}$$

By comparing with Eq. S.46, we can see that, for the equality in Eq S.44 to hold, it is sufficient to show that $\tilde{w}_j^{(t)} = w_j t$ for all j and all t. The kernel mixture weights are uniformly initialized to $w_j^{(0)} = \frac{1}{N}$, and thus equal to the count-based initial weights (Eq. S.48). The kernel mixture weights are updated with the incremental update rule derived above (Eq. S.43), i.e.

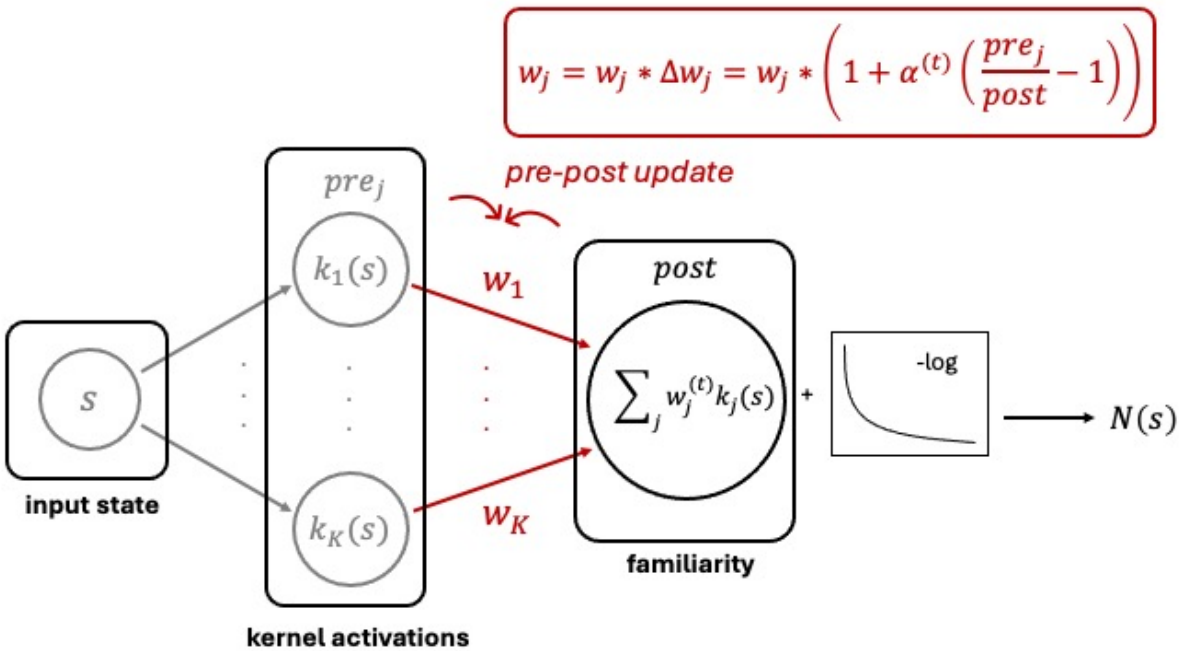
$$\begin{aligned}
w_j^{(t)} &= w_j^{(t-1)} + \frac{1}{t + \varepsilon_k(N-1)} \left(\gamma_{j,t}^{(t)} - w_j^{(t-1)}\right) && \text{(S.56)}\\
&= w_j^{(t-1)} \left(1 - \frac{1}{t + \varepsilon_k(N-1)}\right) + \frac{1}{t + \varepsilon_k(N-1)} \gamma_{j,t}^{(t)} && \text{(S.57)}\\
&= w_j^{(t-1)} \left(1 - \frac{1}{t + \varepsilon_k(N-1)}\right) + \frac{\delta(s_t, s_j)}{t + \varepsilon_k(N-1)}, && \text{(S.58)}
\end{aligned}$$

where we have used that, given our choice of box kernels, we have

$$\gamma_{j,t}^{(t)} = \frac{w_j^{(t)} \delta(s_t, s_j)}{\sum_{j'=1} w_{j'}^{(t)} \delta(s_t, s_{j'})} = \delta(s_t, s_j). \tag{S.59}$$

By comparing with the update rule of the count-based mixture weights (Eq. S.53), we can see that by setting the kernel-based prior $\varepsilon_k = \frac{N}{N-1}$, the two weight updates become equivalent, and, hence, the two novelty definitions are equal in the sense of Eq. S.44.

1.4 Possible network implementation of kernel-based novelty



2 Extraction of neural responses in Homann et al.

Population responses are extracted from the raw fluorescence traces in six steps: (i) fluorescence traces $F(t)$ for each ROI (manually identified) are obtained by averaging over the ROI's pixels on a frame-by-frame basis; (ii) relative fluorescence traces for each ROI are computed as $\Delta F/F(t) = \frac{F(t)-F_0(t)}{F_0(t)}$, where a smooth estimate of the baseline fluorescence $F_0(t)$ is computed using spline interpolation; (iii) event-triggered activity (ETA) traces for each event type (novel image, familiar image, recovery probe) is computed by trial-averaging relative fluorescence traces $\Delta F/F(t)$ in the window 1000 ms before until 300-500 ms after the event, separately for each ROI; (iv) excess activity traces in each ROI for the novel image and the recovery probe events are obtained by subtracting the ETA for familiar image events from the respective ETAs of novel image and recovery probe events; (v) population excess activity traces (and population steady-state activity traces) are computed as the population average over the individual ROI's excess (and steady-state) activity traces; (vi) the population's novelty and recovery probe responses ΔN are given by the maximum amplitude of the population excess activity to the respective event (novel image or recovery probe), while the population's steady-state response N_∞ to familiar images is calculated as the average of the population steady-state activity trace.

3 Fit to neural data: details

3.1 Linear regression fit to neural data

For each parameter set θ of a given model $\mathscr{M}$, we use least-squares estimation (LSE, see below) to find the regression coefficients β that minimize the regression error ε in the regression equation

$$Y = \beta X + \varepsilon, \tag{S.60}$$

where Y is the neural data vector, and X contains the model predictions for Y. The data vector $Y \in \mathbb{R}^{20}$ contains the average neural responses to novelty (and familiarity) measured in the three experiment variations:

$$Y = \left(y_1^{(1)}, ..., y_5^{(1)}, y_1^{(2)}, ..., y_7^{(2)}, y_1^{(3)}, ..., y_4^{(3)}, y_1^{(4)}, ..., y_4^{(4)}\right) \tag{S.61}$$

where $y_i^{(1)}$ are the average neural novelty responses ΔN during the 'variable repetition experiment' for different number of repetitions L of the familiar sequence ($L = 1, 3, 8, 18, 38$); $y_i^{(2)}$ are the average neural novelty responses ΔN to the formerly familiar sequence in the 'repeated image set experiment', for different lengths ΔT of the replacement interval ($\Delta T = 0, 21, 42, 63, 84, 108, 144$ seconds); and $y_i^{(3)}$ and $y_i^{(4)}$ are the average neural responses to novelty, ΔN, and to familiarity, N_∞, respectively, during the 'variable image number experiment' for different number M of familiar stimuli ($M = 3, 6, 9, 12$). The model predictions $X \in \mathbb{R}^{20\times 2}$ are given as

$$X = \begin{pmatrix} 0 & \dots & 0 & 1 & \dots & 1 \\ x_1^{(1)} & \dots & x_4^{(3)} & x_1^{(4)} & \dots & x_4^{(4)} \end{pmatrix}, \tag{S.62}$$

where the row column, containing the model's respective predictions for the novelty and familiarity signals in each of the experimental conditions underlying the data vector Y, is used to fit the scaling factor β_2 (second entry of $\beta \in \mathbb{R}^2$) of the linear regression, and the first row is used to fit the shift β_1 (first entry of $\beta \in \mathbb{R}^2$) of the predicted steady-state responses N_∞ relative to the actual neural signals to familiar stimuli. Note that since the novelty responses ΔN are normalized with the steady-state responses, the shift does not apply to them, such that they have zero-entries in the first row of X.

For each model $\mathscr{M}$, we choose the parameter set θ that minimizes the mean squared error (MSE) between neural data Y and the fitted model predictions βX^*, normalized with the variance of the neural data $\sigma^2(Y)$:

$$\varepsilon_{\text{regr}} = \frac{\text{MSE}(Y, \beta X^*)}{\sigma^2(Y)}. \tag{S.63}$$

Cross-validation. We perform cross-validation of each fitted model $\mathscr{M}$ as follows: we fit $\mathscr{M}$ to the neural data from two out of the three experiments by Homann et al., and compute the cross-validation error as the regression score $\varepsilon_{\text{regr}}$ for the fitted model on the left-out data set. For example, to compute the cross-validation error of model $\mathscr{M}$ on the 'variable repetition experiment', we fit $\mathscr{M}$ to the reduced data vector

$$\tilde{Y}^{(1)} = (y_1^{(2)}, ..., y_7^{(2)}, y_1^{(3)}, ..., y_4^{(3)}, y_1^{(4)}, ..., y_4^{(4)}) \tag{S.64}$$

and compute the cross validation error as the regression error on the left-out data $Y^{(1)} = (y_1^{(1)}, ..., y_5^{(1)})$:

$$\varepsilon_{\text{cross}} = \frac{MSE\left(Y^{(1)}, \beta X^{*(1)}\right)}{\sigma^2\left(Y^{(1)}\right)}, \tag{S.65}$$

where $X^{*(1)}$ contains the predictions of the model fitted to $\tilde{Y}^{(1)}$ for the 'variable repetition experiment'. We report the average cross-validation error for each model across the three cross-validation settings (leaving out data from the 'variable repetition', the 'variable image number' and the 'repeated image set' experiment, respectively).

3.2 Least-squares estimation (LSE)

Least-squares estimation (LSE) is a method to determine the regression coefficients $\beta \in \mathbb{R}^m$ that minimize the regression error $\varepsilon \in \mathbb{R}^n$ in the regression equation

$$Y = \beta X + \varepsilon, \tag{S.66}$$

with dependent variable $Y \in \mathbb{R}^n$ and independent variable $X \in \mathbb{R}^{m \times n}$. LSE computes the determines the regression coefficients as

$$\beta^* = \arg\min_{\beta} \mathscr{L}(X, Y, \beta) = \arg\min_{\beta} |\beta X - Y|^2, \tag{S.67}$$

where the loss $\mathscr{L}(X, Y, \beta)$ can be rewritten as

$$\mathscr{L}(X, Y, \beta) = (\beta X - Y)^T (\beta X - Y) \tag{S.68}$$

$$= \beta^T X^T X \beta - 2X^T \beta Y + Y^T Y. \tag{S.69}$$

To determine the minimum of Eq. S.69, we set the derivative of the loss with respect to β to zero and determine β^* as the solution of the resulting equation

$$\frac{\mathrm{d}\mathscr{L}(X, Y, \beta)}{\mathrm{d}\beta} = -2X^T Y + 2X^T X \beta = 0, \tag{S.70}$$

yielding

$$\beta^* = (X^T X)^{-1} X^T Y. \tag{S.71}$$

Note that this equation has a solution only if $X^T X$ is invertible.

4 Novelty-seeking RL (N-RL) algorithms

4.1 Model-based N-RL algorithm

The model-based (MB) N-RL algorithm (Alg. 1) uses prioritized sweeping (e.g., see [71]) for the model-based update of the novelty-based Q-values ('N-Q-values'), and applies a softmax policy on the N-Q-values to determine its actions.

The MB N-RL algorithm is characterized by the parameters of the novelty model (ε_c for count-based novelty; ε_k and $\mathbf{k}$ for kernel-based novelty, see Update rule for kernel-based novelty), as well as five additional parameters (see line 2 in Alg. 1). The inverse temperature β determines the noise level of the agent's softmax policy over the Q-values (low β leads to more noisy actions). The leak factor k_{leak} and the prior ε_{env} of the belief counts α determine how strongly the α values for a given transition decay between two observations of that transition - the higher k_{leak}, the more influenced are the α values by recently observed transitions, and the lower the influence of the prior ε_{env}. A lower ε_{env}, in turn, causes the α values to adjust more quickly to the observed transition counts. The more deterministic our

Algorithm 1 Model-based N-RL algorithm

Parameter specification
Specify environment parameters S, A, p, and $s_{\mathrm{G}}, T_{\max}$.
Specify algorithm parameters $\{\lambda, T_{\mathrm{PS}}, k_{\mathrm{leak}}, \varepsilon_{\mathrm{env}}, \beta\}$.
Specify novelty parameters $\{\varepsilon_c\}$ (count-based novelty) or $\{\varepsilon_k, \mathbf{k}\}$ (kernel-based novelty).
Variable initialization
Initialize novelty variables using Alg. 5 (count-based novelty) or Alg. 8 (kernel-based novelty).
Initialize $U^{(1)}(s) = \frac{N^{(1)}(s)}{1-\lambda}$ for all $s \in S$.
Initialize Q-values as $Q^{(1)}_{MB}(s,a) = U^{(1)}(s)$ for all $(s,a) \in S \times A$.
Initialize $\alpha^{(1)}_{s,a}(s') = \begin{cases} \varepsilon_{\mathrm{env}} & \text{if } p(s,a,s') > 0 \\ \mathrm{nan} & \text{if } p(s,a,s') = 0 \end{cases}$ for all $(s,s',a) \in S \times S \times A$.
Initialize agent.
Set $t \leftarrow 0$ and initialize state $S_1 = s_1$.
Update novelty variables using Alg. 7 (count-based novelty) or Alg. 10 (kernel-based novelty).
Initialize time $t \leftarrow 1$.
# Simulate agent.
while $S_t \neq s_{\mathrm{G}}$ and $t < T_{\max}$ **do**
# Take action
    Sample action a_t from $\pi(A_t = a | S_t = s) \propto \exp\left(\beta Q^{(t)}_{MB}(s,a)\right)$.
    Evaluate action a_t in environment and observe $S_{t+1} = s_{t+1}$.
Update internal variables
    Update novelty variables using Alg. 7 (count-based novelty) or Alg. 10 (kernel-based novelty).
    Update $\alpha^{(t+1)}_{s,a}(s') = k_{\mathrm{leak}} \alpha^{(t)}_{s,a}(s') + (1 - k_{\mathrm{leak}}) \varepsilon_{\mathrm{env}}$ for all $(s,a,s') \in S \times A \times S$.
    Update $\alpha^{(t+1)}_{s_t,a_t}(s') = \alpha^{(t)}_{s_t,a_t}(s') + \delta(s', s_{t+1})$ for all $s' \in S$.
    Update beliefs $\hat{\theta}^{(t+1)}(s') = \alpha^{(t+1)}_{s,a}(s') / \sum_{\tilde{s}' \in S} \alpha^{(t+1)}_{s,a}(\tilde{s}')$.
    Compute $N^{(t+1)}(s)$ using Alg.6 (count-based novelty) or Alg. 9 (kernel-based novelty)
    Compute new Q-values $Q^{(t+1)}_{MB}(s,a)$ and new U-values $U^{(t+1)}(s)$ for all $(s,a) \in S \times A$, using $N^{(t+1)}(s)$ and Alg. 2
Update time step
    $t \leftarrow t + 1$.
end while

environment is, the lower our $\varepsilon_{\mathrm{env}}$ should thus be, since every observation contains reliable information about the underlying state transition. The parameter λ and T_{PS} characterize the Q-value updates: T_{PS} determines how many sweeps, i.e. Q-value updates are performed in each step (the higher T_{PS}, the more precise the model-based Q-values update), while λ has a similar role to the discount factor γ that discounts the influence of future expected novelty on the state value of a given state s (the higher λ, the more important the value of future novelty).

4.2 Model-free N-RL algorithm

The model-free (MF) N-RL algorithm (Alg. 3) implements an actor-critic algorithm [71] that updates novelty-based V-values using TD-learning and novelty-based action preferences using policy gradient learning. Actions are chosen using a softmax policy over the action preferences.

In addition to the parameters of the novelty model (same as above for the MB N-RL algorithm), the MF N-RL algorithm is characterized by 8 parameters: the inverse temperature β of the softmax action policy, the initial V-values V_0 and action preferences h_0, the learning rates for critic (α_c) and actor (α_a),

Algorithm 2 Prioritized Sweeping with target signal $N^{(t+1)}$

Initialize next Q-values $Q^{(t+1)}(s,a) = \sum_{s' \in S} \hat{\theta}^{(t+1)}_{s,a} s'(N^{(t+1)}(s') + \lambda U^{(t)}(s'))$ for all $(s,a) \in S \times A$.
Initialize next U-values $U^{(t+1)}(s) = U^{(t)}(s)$ for all $s \in S$.
Initialize priority queue $q = \left\{ \left| \max_{a \in A} Q^{(t+1)}(s,a) - U^{(t+1)}(s) \right| \right\}_{s \in S}$.
# Perform prioritized Q-value update
for $\tau = 1, ..., T_{PS}$ **do**
  Determine state with largest U-value update as $s_\tau = \arg\max(q)$.
  Compute U-value update for s_τ as $\Delta U^{(t+1)}_\tau = \max_{a \in A} Q^{(t+1)}(s_\tau, a) - U^{(t+1)}(s_\tau)$.
  Update $U^{(t+1)}(s_\tau) \leftarrow U^{(t+1)}(s_\tau) + \Delta U^{(t+1)}_\tau = \max_{a \in A} Q(s_\tau, a)$.
  Update next Q-values $Q^{(t+1)}(s,a) \leftarrow Q^{(t+1)}(s,a) + \lambda \hat{\theta}^{(t+1)}(s,a,s_\tau) \Delta U^{(t+1)}_\tau$ for all $(s,a) \in S \times A$.
  Recompute priority queue $q \leftarrow \left\{ \left| \max_{a \in A} Q^{(t+1)}(s,a) - U^{(t+1)}(s) \right| \right\}_{s \in S}$.
end for

the discount factor for future novelty γ, and the two decay factors of the critic and actor eligibility traces λ and $\tilde{\lambda}$.

Algorithm 3 Model-free N-RL algorithm

Parameter specification
Specify environment parameters S, A, p, s_G, $T_{\max}$.
Specify algorithm parameters $\{\lambda, \tilde{\lambda}, \gamma, \alpha_c, \alpha_a, \beta, V_0, h_0\}$.
Specify novelty parameters $\{\varepsilon_c\}$ (count-based novelty) or $\{\varepsilon_k, \mathbf{k}\}$ (kernel-based novelty).
Variable initialization
Initialize novelty variables using Alg. 5 (count-based novelty) or Alg. 8 (kernel-based novelty).
Initialize V-values as $V^{(1)}_{MF}(s) = V_0$ for all $s \in S$.
Initialize action preference values as $h^{(1)}_{MF}(s,a) = h_0$ for all $(s,a) \in S \times A$.
Initialize critic eligibility traces as $e^{(1)}(s) = 0$ for all $s \in S$.
Initialize actor eligibility traces as $\tilde{e}^{(1)}(s,a) = 0$ for all $(s,a) \in S \times A$.
Initialize agent.
Set $t \leftarrow 0$ and initialize state $S_1 = s_1$.
Update novelty variables using Alg. 7 (count-based novelty) or Alg. 10 (kernel-based novelty).
Initialize time $t \leftarrow 1$.
# Simulate agent.
while $S_t \neq s_{\mathrm{G}}$ and $t < T_{\max}$ **do**
# Take action
  Sample action a_t from $\pi(A_t = a | S_t = s) \propto \exp\left(\beta h^{(t)}_{MF}(s,a)\right)$.
  Evaluate action a_t in environment and observe $S_{t+1} = s_{t+1}$.
Update internal variables
  Update novelty variables using Alg. 7 (count-based novelty) or Alg. 10 (kernel-based novelty).
  Compute TD-error $\mathrm{TD}^{(t+1)} = N^{(t+1)}(s_{t+1}) - \gamma V^{(t)}_{MF}(s_{t+1}) - V^{(t)}_{MF}(s_t)$.
  Update critic eligibility traces $e^{(t+1)}(s) = \lambda e^{(t)}(s) + \delta(s, s_t)$ for all $s \in S$.
  Update actor eligibility traces $\tilde{e}^{(t+1)}(s,a) = \tilde{\lambda} e^{(t)}(s,a) + \delta(s, s_t)\delta(a, a_t)$ for all $(s,a) \in S \times A$.
  Update V-values $V^{(t+1)}_{MF}(s) = V^{(t)}_{MF}(s) + \alpha_c e^{(t+1)}(s) \mathrm{TD}^{(t+1)}$ for all $s \in S$.
  Update action preferences $h^{(t+1)}_{MF}(s,a) = h^{(t)}_{MF}(s,a) + \alpha_a \tilde{e}^{(t+1)}(s,a) \mathrm{TD}^{(t+1)}$ for all $(s,a) \in S \times A$.
Update time step
  $t \leftarrow t + 1$.
end while

4.3 Hybrid N-RL algorithm

The hybrid N-RL algorithm runs the model-based and the model-free N-RL updates in parallel and uses a hybrid policy that is based on a linear combination of the model-based novelty Q-values and the model-free novelty action preferences.

The hybrid N-RL model has one set of novelty parameters, analogously to the MF and MB N-RL algorithms above, as well as the respective MF and MB algorithm parameters from Algs. 1,3. The only additional parameter is the weight w_{MF} that determines the balance of the MF and MB softmax distribution in the hybrid policy.

Algorithm 4 Hybrid N-RL algorithm

Parameter specification

Specify environment parameters S, A, p, s_G, $T_{\max}$.

Specify algorithm parameters $\{\beta_{MB}, \lambda_{MB}, T_{\text{PS}}, k_{\text{leak}}, \varepsilon_{\text{env}}, \beta_{MF}, \lambda_{MF}, \tilde{\lambda}_{MF}, \gamma, \alpha_c, \alpha_a, V_0, h_0\}$.

Specify novelty parameters $\{\varepsilon_c\}$ (count-based novelty) or $\{\varepsilon_k, \mathbf{k}\}$ (kernel-based novelty).

Variable initialization

Initialize novelty variables using Alg. 5 (count-based novelty) or Alg. 8 (kernel-based novelty).

Model-based variables

Initialize $U^{(1)}(s) = \frac{N^{(1)}(s)}{1-\lambda_{MB}}$ for all $s \in S$.

Initialize model-based Q-values as $Q^{(1)}_{MB}(s,a) = U^{(1)}(s)$ for all $(s,a) \in S \times A$.

Initialize $\alpha^{(1)}_{s,a}(s') = \begin{cases} \varepsilon_{\text{env}} & \text{if } p(s,a,s') > 0 \\ \text{nan} & \text{if } p(s,a,s') = 0 \end{cases}$ for all $(s,s',a) \in S \times S \times A$.

Model-free variables

Initialize model-free V-values as $V^{(1)}_{MF}(s) = V_0$ for all $s \in S$.

Initialize model-free action preference values as $h^{(1)}_{MF}(s,a) = h_0$ for all $(s,a) \in S \times A$.

Initialize critic eligibility traces as $e^{(1)}(s) = 0$ for all $s \in S$.

Initialize actor eligibility traces as $\tilde{e}^{(1)}(s,a) = 0$ for all $(s,a) \in S \times A$.

Initialize agent.

Set $t \leftarrow 0$ and initialize state $S_1 = s_1$.

Update novelty variables using Alg. 7 (count-based novelty) or Alg. 10 (kernel-based novelty).

Initialize time $t \leftarrow 1$.

Simulate agent.

while $S_t \neq s_{\text{G}}$ and $t < T_{\max}$ **do**

Take action using hybrid policy

Sample action a_t from hybrid policy

$$\pi(A_t = a | S_t = s) \propto w_{MF} \exp\left(\beta_{MF} h^{(t)}_{MF}(s,a)\right) + (1 - w_{MF}) \exp\left(\beta_{MB} Q^{(t)}_{MB}(s,a)\right).$$

Evaluate action a_t in environment and observe $S_{t+1} = s_{t+1}$.

Update internal variables

Update novelty variables using Alg. 7 (count-based novelty) or Alg. 10 (kernel-based novelty).

Model-based update

Update $\alpha^{(t+1)}_{s,a}(s') = k_{\text{leak}} \alpha^{(t)}_{s,a}(s') + (1 - k_{\text{leak}}) \varepsilon_{\text{env}}$ for all $(s,a,s') \in S \times A \times S$.

Update $\alpha^{(t+1)}_{s_t,a_t}(s') = \alpha^{(t)}_{s_t,a_t}(s') + \delta(s', s_{t+1})$ for all $s' \in S$.

Update beliefs $\hat{\theta}^{(t+1)}(s') = \alpha^{(t+1)}_{s,a}(s') / \sum_{\tilde{s}' \in S} \alpha^{(t+1)}_{s,a}(\tilde{s}')$.

Compute $N^{(t+1)}(s)$ using Alg.6 (count-based novelty) or Alg. 9 (kernel-based novelty)

Compute new Q-values $Q^{(t+1)}_{MB}(s,a)$ and new U-values $U^{(t+1)}(s)$ for all $(s,a) \in S \times A$, using $N^{(t+1)}(s)$ and Alg. 2

Model-free update

Compute TD-error $\text{TD}^{(t+1)} = N^{(t+1)}(s_{t+1}) - \gamma V^{(t)}_{MF}(s_{t+1}) - V^{(t)}_{MF}(s_t)$.

Update critic eligibility traces $e^{(t+1)}(s) = \lambda_{MF} e^{(t)}(s) + \delta(s, s_t)$ for all $s \in S$.

Update actor eligibility traces $\tilde{e}^{(t+1)}(s,a) = \tilde{\lambda}_{MF} e^{(t)}(s,a) + \delta(s, s_t)\delta(a, a_t)$ for all $(s,a) \in S \times A$.

Update V-values $V^{(t+1)}_{MF}(s) = V^{(t)}_{MF}(s) + \alpha_c e^{(t+1)}(s) \text{TD}^{(t+1)}$ for all $s \in S$.

Update action preferences $h^{(t+1)}_{MF}(s,a) = h^{(t)}_{MF}(s,a) + \alpha_a \tilde{e}^{(t+1)}(s,a) \text{TD}^{(t+1)}$ for all $(s,a) \in S \times A$.

Update time step

$t \leftarrow t + 1$.

end while

Algorithm 5 Novelty initialization (count-based novelty)

Set $C^{(0)}(s) = 0$ for all $s \in S$.

Algorithm 6 Novelty evaluation (count-based novelty)

Compute novelty $N^{(t+1)}(s) = -\log \frac{C^{(t+1)}(s)+\varepsilon}{t+|S|\varepsilon}$ for all $s \in S$.

Algorithm 7 Novelty update (count-based novelty)

Update counts $C^{(t+1)}(s) = C^{(t)}(s) + \delta(s, S_{t+1})$ for all $s \in S$.

Algorithm 8 Novelty initialization (kernel-based novelty)

Set kernel weights $w_j^{(0)} = \frac{1}{K}$ for all $j = 1, ..., K$.

Algorithm 9 Novelty evaluation (kernel-based novelty)

Compute novelty $N^{(t+1)}(s) = -\log \sum_{j=1}^{K} w_j^{(t+1)} k_j(s)$ for all $s \in S$.

Algorithm 10 Novelty update (kernel-based novelty)

Compute responsibilities $\gamma_j^{(t+1)} = \frac{w_j^{(t)} k_j(S_{t+1})}{\sum_{j'=1}^{K} w_{j'}^{(t)} k_{j'}(S_{t+1})}$ for all $j = 1, ..., K$.

Update novelty weights $w_j^{(t+1)} = w_j^{(t)} + \frac{1}{t+K\tilde{\varepsilon}} \left(\gamma_j^{(t+1)} - w_j^{(t)} \right)$.

4.4 Novelty models for N-RL algorithm

5 Behavioural fit: MLE derivation for N-RL models

To compute the log-likelihood in Eq. 25 for a given N-RL model, we rewrite it in terms of the model's softmax policy $\pi(\cdot|\mathscr{M}, \theta)$ (see SI) as

$$\log P(\mathscr{D}|\mathscr{M}, \theta) = \sum_{i=1}^{20} \sum_{t=1}^{T_i} \log \pi(a_{t-1}^{(i)} | s_{t-1}^{(i)}; \mathscr{M}, \theta). \tag{S.72}$$

To compute the right-hand side of Eq. S.72, we simulate the model $\mathscr{M}$ with parameters θ for each mouse trajectory; but, instead of letting the model choose the action in each step t according to its policy, we choose the corresponding action $a_t^{(i)}$ from the mouse data and compute its log-likelihood under the model's policy. These individual log-likelihoods are then added to obtain the total likelihood of the data under the given model $\mathscr{M}$ with parameters θ (Eq. S.72).

The expression for the log-likelihood in Eq. S.72 can be derived as follows:

$$\log P(\mathscr{D}|\mathscr{M},\theta) \tag{S.73}$$

$$= \log \prod_{i=1}^{20} P(s_0^{(i)}, a_0^{(i)} ..., a_{T_1-1}^{(i)}, s_{T_1}^{(i)} | \mathscr{M},\theta) \tag{S.74}$$

$$= \log \prod_{i=1}^{20} P(s_0^{(i)}|\mathscr{M},\theta) P(a_0^{(i)}|s_0^{(i)};\mathscr{M},\theta) P(s_1^{(i)}|a_0^{(i)};\mathscr{M},\theta) ... P(s_{T_1}^{(i)}|a_{T_1-1}^{(i)};\mathscr{M},\theta) \tag{S.75}$$

$$= \sum_{i=1}^{20} \left(\log P(s_0^{(i)}|\mathscr{M},\theta) + \sum_{t=1}^{T_i} \log P(a_{t-1}^{(i)}|s_{t-1}^{(i)};\mathscr{M},\theta) + \sum_{t=1}^{T_i} \log P(s_t^{(i)}|a_{t-1}^{(i)};\mathscr{M},\theta) \right), \tag{S.76}$$

where we have use the Markov property of the N-RL models (see above). Note that $P(s_0^{(i)}|\mathscr{M},\theta) = 1$ since all mice and agents start in the home cage by experiment design; and that $P(s_{t+1}^{(i)}|a_t^{(i)};\mathscr{M},\theta) = 1$ since the environment is deterministic, i.e. given action a always leads to the same state s. Using this, we can simplify Eq. S.76 to

$$\log P(\mathscr{D}|\mathscr{M},\theta) = \sum_{i=1}^{20} \sum_{t=1}^{T_i} \log P(a_{t-1}^{(i)}|s_{t-1}^{(i)};\mathscr{M},\theta). \tag{S.77}$$

We can further write the individual log-likelihoods in Eq. S.77 in terms of the model's action policy $\pi(\cdot|\mathscr{M},\theta)$ (in the case of our N-RL models, a softmax policy over the model's Q-values):

$$\log P(a_{t-1}^{(i)}|s_{t-1}^{(i)};\mathscr{M},\theta) = \log \pi(a_{t-1}^{(i)}|s_{t-1}^{(i)}|\mathscr{M},\theta) \tag{S.78}$$

$$= \log \mathrm{softmax}(Q_{\mathscr{M},\theta}^{(t-1)}(s_{t-1}^{(i)}, a_{t-1}^{(i)})) \tag{S.79}$$

$$= \log \frac{\exp\left(\beta Q_{\mathscr{M},\theta}^{(t-1)}(s_{t-1}^{(i)}, a_{t-1}^{(i)})\right)}{\sum_{a'} \exp\left(\beta Q_{\mathscr{M},\theta}^{(t-1)}(s_{t-1}^{(i)}, a')\right)} \tag{S.80}$$

$$= \beta Q_{\mathscr{M},\theta}^{(t-1)}(s_{t-1}^{(i)}, a_{t-1}^{(i)}) - \log \sum_{a'} \exp\left(\beta Q_{\mathscr{M},\theta}^{(t-1)}(s_{t-1}^{(i)}, a')\right), \tag{S.81}$$

where the inverse temperature β of the softmax policy is part of the set of model parameters θ.

6 Behavioral data by Rosenberg et al.: Outliers

For mouse 'B2', which fails to receive a reward during its first 6 visits to the goal state, we only consider behavior up to its first goal encounter. Like Rosenberg et al., we exclude mouse 'D6' since it fails to enter the maze or more than one short (1 time step) bout. In contrast to Rosenberg et al., we include mouse 'C6' in our analysis since its behavior with respect to our behavioral statistic does not show outlier behavior. While Rosenberg et al. analyze exploration behavior based on 'first water runs', which includes the direct paths between maze entry and the goal state in the maze, we analyze the entire state trajectory of mice during the exploration between first entry and encounter of the goal state.

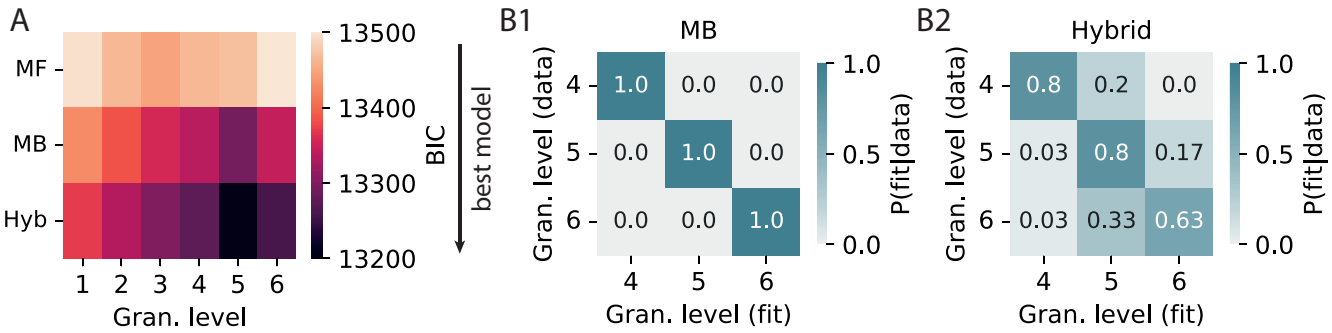


Figure 6: Model recovery for all kernel-based novelty models in the Rosenberg task.